Journal of Computer Security 00 (2006) 1–41 1
IOS Press

On the security of group communication schemes

Shouhuai Xu *
Department of Computer Science, University of Texas at San Antonio, USA
E-mail: shxu@cs.utsa.edu

Secure group communications are a mechanism facilitating protected transmission of messages from a sender to multiple receivers, and many emerging applications in both wired and wireless networks need the support of such a mechanism. There have been many secure group communication schemes in wired networks, which can be directly adopted in, or appropriately adapted to, wireless networks such as mobile ad hoc networks (MANETs) and sensor networks. In this paper we show that the popular group communication schemes that we have examined are vulnerable to the following attack: An outside adversary who compromises a certain legitimate group member could obtain *all* past and present group keys (and thus all the messages protected by them); this is in sharp contrast to the widely-accepted belief that a such adversary can only obtain the present group key (and thus the messages protected by it). In order to understand and deal with the attack, we formalize two security models for stateful and stateless group communication schemes. We show that some practical methods can make a *subclass* of existing group communication schemes immune to the attack.

Keywords: Security, key management, group communication, multicast

1. Introduction

Secure group communications are useful in both wired and wireless networks, because they facilitate protected transmission of messages from a sender to multiple receivers. One important property of secure group communications is to ensure that only the legitimate members (or users, receivers) can have access to the multicast or broadcast data. There have been many secure group communication schemes in the setting of wired networks; popular ones include the stateful LKH [27,28] and OFT [2,25] as well as stateless ones [12,22]. These schemes can be directly adopted in, or appropriately adapted to, the setting of wireless networks such as mobile ad hoc networks (MANETs) and sensor networks. The core component of a secure group communication scheme is its key management method. A common feature among these schemes' key management methods is that each user holds a set of keys that are then utilized to help establish some group keys (which are common to all the group members and are used to encrypt actual messages).

In this paper we show that these group communication schemes, or more specifically their key management methods, are subject to the following attack: An outside

*An extended abstract of this paper appeared as [29].

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 2

adversary who compromises a certain legitimate group member could obtain *all* past and present group keys (and thus the data encrypted using these keys). This is in sharp contrast to the widely-accepted belief that such an adversary can only obtain the present group key. This attack is powerful also because it provides the adversary the following flexibility: There are potentially many legitimate group members such that compromising any (or a small number) of them leads to the exposure of both past and present group keys. This flexibility may be particularly relevant in the setting of MANETs and sensor networks because they are typically deployed in a small area and the adversary can capture and compromise the easiest-to-obtain node(s).

1.1. Motivating problems

Now we explore some attack scenarios against the stateful LKH [27,28] and OFT [2,25], and against stateless ones [12,22]. The emphasis is on the case of LKH.

Vulnerability of the LKH and LKH+ schemes: Let's first briefly review the LKH group communication scheme. Following the notations of [28], we let

$$x \to \{y_1, \ldots, y_\ell\} : \{z\}_w$$

denote that x sends the users $y_1, \ldots, y_\ell$ (via multicast or unicast) the encryption of plaintext z using key w, namely the ciphertext $\{z\}_w$.

Consider the simple scenario, as shown in Fig. 1(a), of a group consisting of a key server s and users $u_1, \ldots, u_8$. The server is responsible for initiating and maintaining the group in the presence of user dynamics (i.e., joins and leaves). The keys are organized as a *key tree*, where the leaves are the users and the inner nodes are the keys. Moreover, each user holds the keys corresponding to the inner nodes on the path starting from the parent of the user and ending at the root. For example, in Fig. 1(a), user u_1 holds keys k_1, k_{123}, and k_{1-8}, where k_{1-8} is the *group key* that can be used to encrypt the communications within the group.

In order to maintain secure communications, each join or leave would require the key server to change some keys that also need to be securely distributed to certain users (via some *rekeying messages*). Ignoring for a moment certain details such as authorization of joining the group and authentication of the messages sent by the key server, in what follows we explain how the key server responds to group dynamics.

After granting a join request from user u_9, server s shares a key k_9 with user u_9. Certain keys need to be changed and sent to certain relevant users. As shown in Fig. 1(b), in order to prevent u_9 from accessing past communications, k_{78} and k_{1-8} are changed to k_{789} and k_{1-9}, respectively. Moreover, the new group key k_{1-9} needs to be securely sent to users $u_1, \ldots, u_9$, and k_{789} needs to be securely sent to users u_7, u_8, and u_9. One efficient way to do this is the following algorithm (which corresponds to the so-called *group-oriented rekeying* strategy):

$$s \to \{u_1, \ldots, u_8\} : \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}}$$
$$s \to \{u_9\} : \{k_{1-9}, k_{789}\}_{k_9}$$

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 3

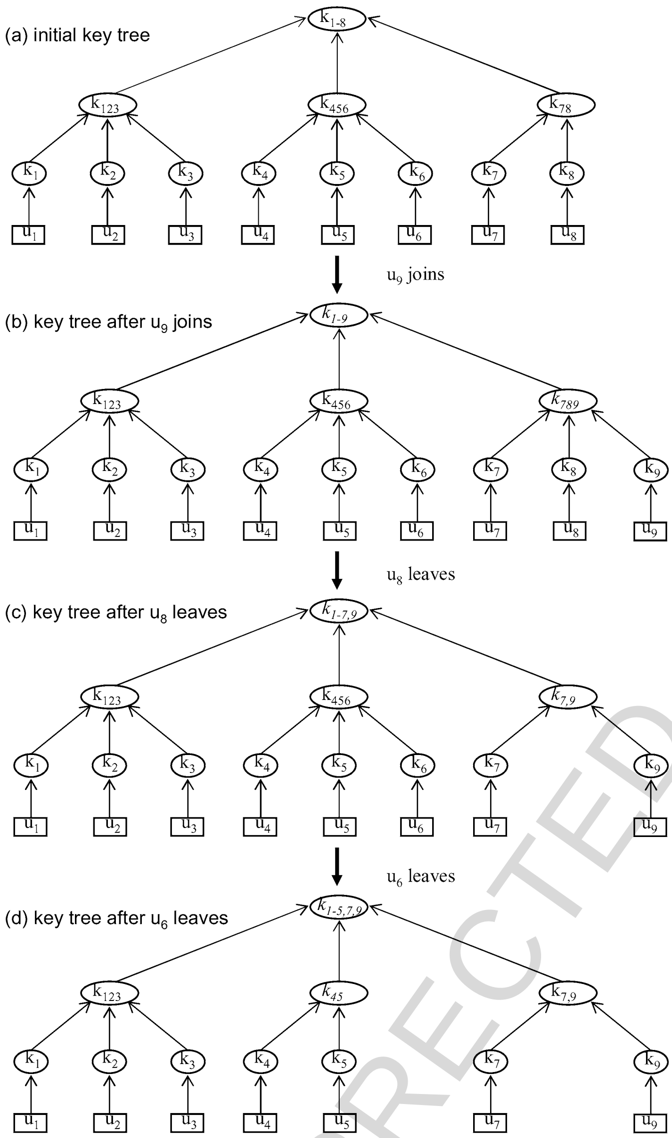

Fig. 1. A scenario of LKH.

Furthermore, k_{1-8} is securely erased by $u_1, \ldots, u_8$, and k_{78} is securely erased by u_7 and u_8.

Now suppose u_8 leaves. To prevent u_8 from accessing future communications, as shown in Fig. 1(c), server s needs to change k_{1-9} and k_{789} to $k_{1-7,9}$ and $k_{7,9}$, respectively. Moreover, the new group key $k_{1-7,9}$ needs to be securely sent to users $u_1, \ldots, u_7, u_9$, and $k_{7,9}$ needs to be securely sent to u_7 and u_9. One efficient way to do this is the following algorithm (which also corresponds to the group-oriented rekeying strategy):

$$s \rightarrow \{u_1, \ldots, u_7, u_9\} : \{k_{1-7,9}\}_{k_{123}}, \{k_{1-7,9}\}_{k_{456}}, \{k_{1-7,9}\}_{k_{7,9}}, \\ \{k_{7,9}\}_{k_7}, \{k_{7,9}\}_{k_9}.$$

Furthermore, k_{1-9} is securely erased by $u_1, \ldots, u_7, u_9$, and k_{789} is securely erased by u_7 and u_9.

Now suppose u_6 leaves also. To prevent u_6 from accessing future communications, as shown in Fig. 1(d), server s needs to change $k_{1-7,9}$ and k_{456} to $k_{1-5,7,9}$ and k_{45}, respectively. Moreover, the new group key $k_{1-5,7,9}$ needs to be securely sent to users $u_1, \ldots, u_5, u_7, u_9$, and k_{45} needs to be securely sent to users u_4 and u_5. One efficient way to do this is the following algorithm (which also corresponds to the group-oriented rekeying strategy):

$$s \rightarrow \{u_1, \ldots, u_5, u_7, u_9\} : \{k_{1-5,7,9}\}_{k_{123}}, \{k_{1-5,7,9}\}_{k_{45}}, \{k_{1-5,7,9}\}_{k_{7,9}}, \\ \{k_{45}\}_{k_4}, \{k_{45}\}_{k_5}.$$

Furthermore, $k_{1-7,9}$ is securely erased by $u_1, \ldots, u_5, u_7, u_9$, and k_{456} is securely erased by u_4 and u_5.

Given the above system setting, let us now examine the consequences of a legitimate user being compromised.

- Suppose an adversary compromises user u_9. It is of course true that the adversary is able to obtain the present group key $k_{1-5,7,9}$, no matter how the group rekeying scheme works. Moreover, the adversary can obtain $k_{7,9}$ and k_9. We observe that the adversary who has recorded the network traffic is also able to obtain the past group keys k_{1-9} and $k_{1-7,9}$, because it can decrypt the messages incurred by the events that u_9 joins the group and that u_8 leaves the group:

$$\begin{aligned} s \rightarrow \{u_9\} \quad & : \{k_{1-9}, k_{789}\}_{k_9}, \\ s \rightarrow \{u_1, \ldots, u_7, u_9\} \quad & : \{k_{1-7,9}\}_{k_{123}}, \{k_{1-7,9}\}_{k_{456}}, \\ & \quad \{k_{1-7,9}\}_{k_{7,9}}, \{k_{7,9}\}_{k_7}, \{k_{7,9}\}_{k_9}. \end{aligned}$$

  We stress that this is true even though the past group keys $k_{1,9}$ and $k_{1-7,9}$ were securely erased by u_9. As a consequence, the adversary can decrypt the communications encrypted using the past and present group keys k_{1-9}, $k_{1-7,9}$, and $k_{1-5,7,9}$. We notice that the initial group key k_{1-8} is never accessible to u_9.
- Suppose an adversary compromises user u_7. Then, the adversary knows $k_{1-5,7,9}$, $k_{7,9}$, and k_7. Note that the adversary can obtain $k_{1-7,9}$ from the

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 5

recorded traffic corresponding to the event that u_8 leaves the group:

$$s \to \{u_1, \ldots, u_7, u_9\} : \{k_{1-7,9}\}_{k_{123}}, \{k_{1-7,9}\}_{k_{456}}, \{k_{1-7,9}\}_{k_{7,9}}, \{k_{7,9}\}_{k_7}, \{k_{7,9}\}_{k_9}.$$

We stress that this is true even though the past group key $k_{1-7,9}$ was securely erased by u_7. We notice that the above analysis is based on the implicit assumption that the initial group key k_{1-8} was "magically" sent to u_7. In practice, k_{1-8} might have been sent to u_7 via its individual key k_7. This means that the adversary can obtain k_{1-8}, and thus k_{1-9} through the recorded traffic corresponding to the event that u_9 joins the group:

$$s \to \{u_1, \ldots, u_8\} : \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}}.$$

As a consequence, *all* past and present group keys, namely k_{1-8}, k_{1-9}, $k_{1-7,9}$ and $k_{1-5,7,9}$, are compromised even if the first three were securely erased by u_7.

- Suppose u_5 is compromised. Then, the adversary knows $k_{1-5,7,9}$, k_{45}, and k_5. Further, if k_{1-8} was sent to u_5 through an encryption using its individual key k_5, then k_{1-8} is exposed. Moreover, k_{1-9} can be obtained by the adversary from the recorded traffic corresponding to the event that u_9 joins the group:

$$s \to \{u_1, \ldots, u_8\} : \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}}.$$

As a consequence, the past and present group keys, namely k_{1-8}, k_{1-9} and $k_{1-5,7,9}$ are compromised, even if they were securely erased by u_5.
A similar analysis applies to the case that u_4 is compromised.

- Suppose u_1 is compromised. Then $k_{1-5,7,9}$, k_{123}, and k_1 are obtained by the adversary. This means that the adversary can further obtain $k_{1-7,9}$ from the recorded traffic corresponding to the event that u_8 leaves the group:

$$s \to \{u_1, \ldots, u_7, u_9\} : \{k_{1-7,9}\}_{k_{123}}, \{k_{1-7,9}\}_{k_{456}}, \{k_{1-7,9}\}_{k_{7,9}}, \{k_{7,9}\}_{k_7}, \{k_{7,9}\}_{k_9}.$$

Further, the above analysis is based on the implicit assumption that the initial group key k_{1-8} was "magically" sent to u_1. In practice, k_{1-8} might have been sent to u_1 via its individual key k_1. This means that the adversary can obtain k_{1-8}, and thus k_{1-9} through the recorded traffic corresponding to the event that u_9 joins the group:

$$s \to \{u_1, \ldots, u_8\} : \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}}.$$

As a consequence, *all* past and present group keys, namely k_{1-8}, k_{1-9}, $k_{1-7,9}$ and $k_{1-5,7,9}$, are compromised even if the first three were securely erased by u_1.
A similar analysis applies to the case that u_2 or u_3 is compromised.

In summary, the above discussion shows, in sharp contrast to the desired property that the adversary can only obtain the present group key $k_{1-5,7,9}$, that compromising any of u_1, u_2, u_3, u_7 could lead to the exposure of all past and present group keys, and compromising any of u_4, u_5, u_9 leads to the exposure of most past and present group keys. This means that the adversary has considerable flexibility in selecting the *weakest* node(s) to compromise. Finally, we remark that the attack is not fundamentally related to the group-oriented rekeying strategy, and that LKH+ [26], which was seemingly motivated from an efficiency perspective, resolves only a piece of the problem because the above attack remains effective when the group dynamics are incurred by leaving events.

Remark 1. While there could be other methods to bootstrap the initial keys (e.g., k_{1-8} is not protected by k_7), the following scenario would still support the above conclusion. Suppose at system initialization there is no user but the server, then users join the system one by one via LKH's join protocol (cf. Appendix A). In this case, transmission of group keys is always protected by individual keys, meaning that compromise of some user (or users) could lead to the exposure of all past and present group keys.

Remark 2. One may observe that the compromise of past group keys may not be a serious problem. This is so because if a node stored all the past communication content, it will be leaked to the adversary when the node is compromised. However, there are situations, such as sensitive applications, where the nodes do not, or even are not allowed to, store past communication content. We notice that this issue is also relevant to [4,7].

Vulnerability of the One-way Function Tree (OFT) scheme: OFT [2,25] is a stateful group communication scheme. The basic idea underlying the OFT scheme is the following (we refer the reader to [2,25] for details). The center maintains a binary tree, each node x of which is associated with two cryptographic keys: a node key k_x and a blinded node key $k'_x = g(k_x)$, where g is an appropriate one-way function. Every leaf of the tree is associated with a group member, and the center assigns a randomly chosen key k_x to each member x. Let f be a "mixing" function (e.g., $\oplus$). The interior node keys are defined by the rule

$$k_u = f(g(k_{\mathit{left}(u)}), g(k_{\mathit{right}(u)}))$$

where *left*(u) and *right*(u) are the left and right children of the node u, respectively. This way, the node key associated with the root of the tree is the group key. In order for a member u to derive the group key, the center (or server, sender) sends the blinded node keys of nodes adjacent to the nodes on (i.e., of the nodes "hanging" off) the path from u to the root.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 7

When a new member joins the group, an existing leaf node u is split, the member associated with u is now associated with *left*(u), and the new member is associated with *right*(u). Both members are given new keys. The new blinded node keys that have been changed are *securely sent* to the appropriate subgroups of members.

When the member associated with a node u is evicted from the group, the member assigned to the sibling of u is reassigned to the parent p of u and given a new leaf key. If the sibling s of u is the root of a subtree, then p becomes s, moving the subtree closer to the root, and one of the leaves of this subtree is given a new key. The new blinded node keys are *securely sent* to the appropriate subgroups of members.

Now we show why the OFT scheme is also vulnerable to a similar attack. The key observation is that whenever there is a change to any blinded node key, the center needs to *securely send* the new blinded node keys to certain other legitimate nodes. It seems that any reasonable method would be based on the keys possessed by the respective nodes (e.g., u). Since u can derive the new group key after receiving the rekeying message, an *outsider* adversary could use the following strategy to recover the group key: it first records the rekeying message, and then breaks into u's computer *after* the next rekeying event (assuming that u is still legitimate). Moreover, compromising any of the nodes that have not been evicted enables the adversary to recover past and present group keys.

Vulnerability of the stateless subset-cover framework: Naor et al. [22] presented the first practical stateless group communication scheme, which has its roots in broadcast encryption [9]. Compared with the stateful group communication schemes discussed above, stateless schemes have the nice feature that they do not assume the receivers (or users, members) being always on-line. Since the receivers do not necessarily update their state from session to session, stateless schemes are especially good for applications over lossy channels (e.g., MANETs and sensor networks). We stress that the security analysis presented in [22] remains sound in its adversarial model; whereas the present paper considers a strictly stronger adversarial model.

The subset-cover framework of [22] is reviewed in Fig. 2, where $\mathcal{N}$ is the set of all users, $\mathcal{R} \subset \mathcal{N}$ is a group of $|\mathcal{R}| = r$ users whose decryption privileges should be revoked, and E_L and F_K are two appropriate symmetric key cryptosystems (whose properties will be specified later). The goal of a stateless group communication scheme is to allow a center to transmit a message M to all users such that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ cannot decrypt it. Suppose $S_1, \ldots, S_w$ are a collection of subsets of users, where $S_j \subseteq \mathcal{N}$ for $1 \leqslant j \leqslant w$, and each S_j is assigned a long-lived key L_j such that each $u \in S_j$ should be able to deduce L_j from its secret information I_u. Given a revoked set $\mathcal{R}$, if one can partition $\mathcal{N} \setminus \mathcal{R}$ into (ideally disjoint) sets $S_{i_1}, \ldots, S_{i_m}$ such that $\mathcal{N} \setminus \mathcal{R} \subseteq \cup_{\ell=1}^{m} S_{i_\ell}$, then a session key K can be encrypted m times with $L_{i_1}, \ldots, L_{i_m}$, and each user $u \in \mathcal{N} \setminus \mathcal{R}$ can obtain K and thus M.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 8

Initialization: Every receiver u is assigned private information I_u. For all $1 \leqslant i \leqslant w$ such that $u \in S_i$, I_u allows u to deduce the key L_i corresponding to the set S_i.

Broadcasting: Given a set $\mathcal{R}$ of revoked receivers, the center (or server, group controller, sender) executes the following:

1. Choose a session encryption key K.
2. Find a partition of the users in $\mathcal{N} \setminus \mathcal{R}$ into disjoint subsets $S_{i_1}, \ldots, S_{i_m}$. Let $L_{i_1}, \ldots, L_{i_m}$ be the keys associated with the above subsets.
3. Encrypt K with keys $L_{i_1}, \ldots, L_{i_m}$ and send the ciphertext

$$\langle [i_1, \ldots, i_m, E_{L_{i_1}}(K), \ldots, E_{L_{i_m}}(K)], F_K(M) \rangle.$$

Decryption: A receiver u, upon receiving a broadcast message $\langle [i_1, \ldots, i_m, C_1, \ldots, C_m], C \rangle$, executes as follows.

1. Find i_j such that $u \in S_{i_j}$ (in the case $u \in \mathcal{R}$ the result is NULL).
2. Extract the corresponding key L_{i_j} from I_u.
3. Decrypt C_j using key L_{i_j} to obtain K.
4. Decrypt C using key K to obtain the message M.

Fig. 2. The subset-cover revocation framework.

The subset-cover framework has a vulnerability similar to the one against the stateful group schemes. Specifically, suppose an adversary $\mathcal{A} \notin \mathcal{N}$ records all the encrypted communications over the channels. If $\mathcal{A}$ breaks into a legitimate user $u \in \mathcal{N}$ at a later point in time, then $\mathcal{A}$ obtains I_u, which allows it to recover the L_{i_j} (and thus the encrypted M) that u was entitled to obtain. In the extreme case that u has never been revoked, $\mathcal{A}$ can recover all past and present keys.

1.2. *Our contributions*

We trace the above vulnerability of group communication schemes back to that their security models (if any) are not sufficient. We formalize two adversarial models. One is called the *passive attack model*, in which the adversary is passive in the sense that it is only allowed to join and leave the group in an arbitrary fashion, but not allowed to corrupt any legitimate members. This model has seemingly been implicitly adopted in the existing group communication literature. The other more realistic one is called the *active outsider attack model*, in which the adversary is further allowed to corrupt legitimate members. This model aids understanding and dealing with the aforementioned attack. In each of the two models, we define two security notions, namely *forward-security* meaning that the revoked or evicted members, even if they collude, cannot obtain the future group keys, and *backward-security* mean-

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 9

ing that a newly admitted member cannot obtain the past group keys.[1] This allows us to obtain the following interesting results about the relationships between these security notions, which are equally applicable to both stateful and stateless group communication schemes (see Sections 3.3 and 6.2, respectively).

1. In the active outsider attack model, backward-security (also called `strong-security`) is *strictly* stronger than forward-security (also called `security`). This means that in the active outsider attack model one only needs to prove the backward-security property.
2. In the passive attack model, backward-security is equivalent to forward-security.
3. Backward-security in the active outsider attack model (i.e., `strong-security`) is *strictly* stronger than backward-security in the passive attack model. However, we do *not* know whether forward-security in the active outsider attack model is also *strictly* stronger than its counterpart in the passive attack model (we only know that when the adversary is *static* they are equivalent).
4. The security achieved in existing group communication schemes (e.g., [12, 22,25,28]) is indeed, as we will show, forward-security in the active outsider attack model (i.e., `security`). This has not become clear until now because there were no formal models specified before (in spite of the fact that the passive attack model has seemingly been implicitly adopted in the literature). The achieved `security` property is at least as strong as what we call backward-security in the passive attack model, but *strictly* weaker than what we call backward-security in the active outsider attack model (i.e., `strong-security`) – a property that blocks the attack discussed above.

Besides the above general results, we show that some practical methods can transform a *subclass* of the group communication schemes (including LKH [27,28], LKH+ [26], and the complete subtree method [22]) into ones that achieve the desired `strong-security`. The methods are based on two general compilers. The extra complexity imposed by the compilers is typically that at each rekeying event a group member conducts logarithmically-many pseudorandom function evaluations. This should not jeopardize their utility even in the setting of MANETs and sensor networks, as pseudorandom functions may be implemented with block ciphers in practice. We also present instantiations of the compilers, which lead to concrete schemes that achieve the desired `strong-security`.

Although the technical means underlying the transformation is to evolve the keys based on a secure pseudorandom function family – an idea inspired by [4], there are some subtle issues in our settings. First, we must allow the adversary to corrupt some group keys that are used to encrypt the communications *before* the rekeying

[1] The terms follow the group communication literature (see, e.g., [27,28]). Their meanings are indeed different from the ones adopted in the cryptographic literature [1,3,4].

message of interest. Of course, the corrupt members must have been revoked before that rekeying message. On the other hand, in [4] no such corruption is allowed before the event of interest. Second, from an adversary's perspective, there could be many "valuable" users in our settings, meaning that an adversary only needs to compromise the *weakest* one(s) of them. Whereas, no such flexibility is given to the adversary in the setting of [4].

1.3. Related work

LKH was independently invented in [27,28]. Although these schemes are mainly invented for secure multicast applications, we believe that many other applications can utilize such a scheme; we refer the reader to [6,23] for a survey, including the relationship between the schemes of [27,28] and the schemes of [9,22]. We notice that the stateless schemes (e.g., [12,22]) are perhaps more useful in an environment of lossy channels. Although the LKH scheme has been extended in several directions, these extensions are motivated to improve performance rather than to achieve strictly stronger security. For example, performance can be improved by periodic group rekeying [24] or batch rekeying [19], and improved trade-offs between storage and communication are available in [6,8,21]. Nevertheless, these techniques may also be utilized in our `strongly-secure` group communication schemes. To the best of our knowledge, our work is the first one that identifies a new and realistic attack, and presents solutions for (a subclass of) the popular group communication schemes. The variant presented in [26] (which is also known as LKH+) is similar to our performance optimization in that the communication complexity incurred by joining events can be substantially reduced. However, there was no formal treatment of the utilized key evolvement, nor was their scheme resistant against the attack introduced in Section 1.1.

While secure group-oriented communications have been extensively investigated in the setting of wired networks, their counterparts in the setting of wireless networks have yet to be thoroughly explored. Although the aforementioned schemes can be directly deployed in wireless settings, a simple-minded adoption may not lead to the desired performance (see, e.g., [17,31]). Fortunately, there have been some interesting investigations that show that these schemes can be adapted (e.g., by taking into account some physical characteristics of ad hoc networks [15–18]) so that better performance can be achieved. One of the practical values of the present paper is that the significantly improved security guarantee in the popular group communication schemes can be seamlessly integrated into the methods for improving performance [15–18]. Indeed, our schemes can be easily integrated into any other methods for improving performance to achieve better security, as long as the methods assume "black-box" access to an underlying security group communication scheme. There have been a few other attempts at securing group communications in such settings: [14] presented a scheme for secure multicast communications in MANETs based on public key cryptosystems; [31] investigated a different approach to secure group communications.

A similar study on enhancing security of public key cryptography based broadcast encryption was investigated in [30].

1.4. Outline

The rest of the paper is organized as follows. In Section 2 we briefly review some cryptographic preliminaries. In Section 3 we present formal models and security definitions of stateful group communication schemes, as well as the relationships between the security notions. In Section 4 we present a compiler for stateful group communication schemes and investigate its properties. The compiler is utilized in Section 5 to derive a concrete `strongly-secure` stateful group communication scheme from the merely `secure` LKH, which is reviewed in Appendix A for completeness. In Section 6 we explore stateless group communication schemes in parallel to their stateful counterparts. We conclude the paper in Section 7.

2. Cryptographic preliminaries

A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if $\forall c\ \exists \kappa_c$ s.t. $\forall \kappa > \kappa_c$, we have $\epsilon(\kappa) < 1/\kappa^c$.

We will base security of group communication schemes on the security of pseudorandom function families. For a security parameter κ, a pseudorandom function (PRF) family $\{f_k\}$ parameterized by a secret value $k \in_R \{0,1\}^\kappa$ has the following property [10]: A probabilistic polynomial-time adversary $\mathcal{A}$ has only a negligible (in κ) advantage in distinguishing f_k from a perfect random function (with the same domain and range). It is well-known that pseudorandom functions can be naturally used to construct symmetric key encryption schemes that are secure against chosen-plaintext attacks (which suffices for our treatment of LKH). Informally, this means that no adversary is able to learn any significant information about the encrypted content. We refer the reader to [13] for a thorough treatment on this subject.

Definition 2.1 (computational independence). Consider a set $S = \{s_1, \dots, s_\ell\}$ of secret binary strings of length κ, where $\ell = \mathit{poly}(\kappa)$ for some polynomial *poly*. We say $s_1, \dots, s_\ell$ are computationally independent of each other if for any probabilistic polynomial-time algorithm $\mathcal{A}$,

$$|\Pr[\mathcal{A}(s_1, \dots, s_\ell) \text{ returns "real"}] - \Pr[\mathcal{A}(r_1, \dots, r_\ell) \text{ returns "real"}]| = \epsilon(\kappa)$$

where $r_1, \dots, r_\ell \in_R \{0,1\}^\kappa$ are uniformly drawn at random, and ϵ is a negligible function.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 12

3. Model and definition of stateful secure group communications

In Section 3.1 we present a formal security model for stateful (and symmetric key cryptography based) group communications. In Section 3.2 we specify the adversarial models and desired security properties. In Section 3.3 we explore the relationships between the security notions.

3.1. Model

Let κ be a security parameter, and $\mathbb{ID}$ be the set of possible group members (i.e., users, receivers, or principals) such that $|\mathbb{ID}|$ is polynomially-bounded in κ. There is a special entity called a *Group Controller* (i.e., key server, center, server, or sender), denoted by $\mathcal{GC}$, such that $\mathcal{GC} \notin \mathbb{ID}$.

Since a stateful group communication scheme is driven by "rekeying" events (because of joining or leaving operations below), it is convenient to treat the events as occurring at "virtual time" $t = 0, 1, 2, \ldots$, because the group controller is able to maintain such an execution history. This indeed accommodates the following important two cases: (1) all the parties periodically update their keys, even if there are no joining or leaving operations – this is relevant when a scheme achieves what we call `strong-security` specified below; (2) the lengths of the time periods do not have to be the same – this is the case when the rekeying events occur in an arbitrary fashion. At time t, let $\Delta^{(t)}$ denote the set of legitimate group members, $k^{(t)} = k^{(t)}_{\mathcal{GC}} = k^{(t)}_{U_1} = \cdots$ the group key,[2] $K^{(t)}_{\mathcal{GC}}$ the set of keys held by the $\mathcal{GC}$, $K^{(t)}_U$ the set of keys held by $U \in \Delta^{(t)}$, and $\mathsf{acc}^{(t)}_U$ the state indicating whether $U \in \Delta^{(t)}$ has successfully received the rekeying message. Initially, $\forall\ U \in \mathbb{ID}, t \in \mathbb{N}$, set $\mathsf{acc}^{(t)}_U \leftarrow \text{FALSE}$. We assume that the $\mathcal{GC}$ treats joining and leaving operation separately (e.g., first fulfilling the leaving operation and then immediately the joining one), even if the requests are made simultaneously. This strategy has indeed been adopted in the group communication literature.

To simplify the presentation, we assume that during the system initialization (i.e., Setup below) or the admission of a joining user, the $\mathcal{GC}$ can communicate with each legitimate member $U \in \mathbb{ID}$ through an *authenticated private* channel, and that after the system initialization the $\mathcal{GC}$ can communicate with any U through an *authenticated* channel. We notice that authenticated channels can by implemented by a digital signature scheme [28], and digital signatures are sometimes necessary [5].

We will not make any *synchronization* assumption about the underlying communication model, which could therefore be asynchronous [20]. However, known practical schemes (e.g., [8,27,28]) assume *reliable* delivery, which would require some (loose) clock synchronization.

[2] It is also known as a session key in the group communication literature.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 13

A group communication scheme has the following components:

Setup: The group controller $\mathcal{GC}$ generates a set of keys $K_{\mathcal{GC}}^{(0)}$, and distributes appropriate subsets of $K_{\mathcal{GC}}^{(0)}$ to the present group members (that may be determined by the adversary), $\Delta^{(0)} \subseteq \mathbb{ID}$, through the authenticated private channels. Each member $U_i \in \Delta^{(0)}$ holds a set of keys denoted by $K_{U_i}^{(0)} \subset K_{\mathcal{GC}}^{(0)}$, and there is a key, $k^{(0)}$ that is common to all the present members, namely $k^{(0)} \in K_{\mathcal{GC}}^{(0)} \cap K_{U_1}^{(0)} \cap \cdots \cap K_{U_{|\Delta^{(0)}|}}^{(0)}$.

Join: This algorithm is executed by group controller $\mathcal{GC}$ at time, say, t due to some join request(s) (we abstract away the out-of-band authentication and establishment of an individual key for each of the new members). It takes as input: (1) identities of previous group members, $\Delta^{(t-1)}$, (2) identities of newly admitted group members, $\Delta' \subseteq \mathbb{ID} \setminus \Delta^{(t-1)}$, (3) keys held by the group controller, $K_{\mathcal{GC}}^{(t-1)}$, and (4) keys held by group members, $\{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}} = \{K_{U_i}^{(t-1)} : U_i \in \Delta^{(t-1)}\}$.

It outputs updated system state information, including: (1) identities of new group members, $\Delta^{(t)} \leftarrow \Delta^{(t-1)} \cup \Delta'$, (2) new keys for the $\mathcal{GC}$ itself, $K_{\mathcal{GC}}^{(t)}$, (3) new keys for new group members, $\{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}$, which are sent to the legitimate users through the authenticated channels, (4) new group key $k^{(t)} \in K_{\mathcal{GC}}^{(t)} \cap K_{U_1}^{(t)} \cap \cdots \cap K_{U_{|\Delta^{(t)}|}}^{(t)}$.

Formally, denote it by $(\Delta^{(t)}, K_{\mathcal{GC}}^{(t)}, \{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}) \leftarrow \mathsf{Join}(\Delta^{(t-1)}, \Delta', K_{\mathcal{GC}}^{(t-1)}, \{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}})$.

Leave: This algorithm is executed by the group controller $\mathcal{GC}$ at time, say, t due to leave or revocation operation(s). It takes as input: (1) identities of previous group members, $\Delta^{(t-1)}$, (2) identities of leaving group members, $\Delta' \subseteq \Delta^{(t-1)}$, (3) keys held by the controller, $K_{\mathcal{GC}}^{(t-1)}$, and (4) keys held by group members, $\{K_{U_i}\}_{U_i \in \Delta^{(t-1)}}^{(t-1)}$.

It outputs updated system state information, including: (1) identities of new group members, $\Delta^{(t)} \leftarrow \Delta^{(t-1)} \setminus \Delta'$, (2) new keys for $\mathcal{GC}$, $K_{\mathcal{GC}}^{(t)}$, (3) new keys for new group members, $\{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}$, which are sent to the legitimate users through the authenticated channels, (4) a new group key $k^{(t)} \in K_{\mathcal{GC}}^{(t)} \cap K_{U_1}^{(t)} \cap \cdots \cap K_{U_{|\Delta^{(t)}|}}^{(t)}$.

Formally, denote it by $(\Delta^{(t)}, K_{\mathcal{GC}}^{(t)}, \{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}) \leftarrow \mathsf{Leave}(\Delta^{(t-1)}, \Delta', K_{\mathcal{GC}}^{(t-1)}, \{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}})$.

Rekey: This algorithm is executed by the legitimate group members belonging to $\Delta^{(t)}$ at time t, where $\Delta^{(t)}$ is derived from a Join or Leave event. Specifically, $U_i \in \Delta^{(t)}$ runs this algorithm upon receiving the message from the $\mathcal{GC}$ over the

authenticated channel. The algorithm takes as input the received message and U_i's secrets, and is supposed to output the updated keys for the group member. If the execution of the algorithm is successful, U_i sets: (1) $\mathsf{acc}_{U_i}^{(t)} \leftarrow \text{TRUE}$, (2) $K_{U_i}^{(t)}$, where $k_{U_i}^{(t)} \in K_{U_i}^{(t)}$ is supposed to be the new group key.
If the rekeying event is incurred by a Join event, every $U_i \in \Delta^{(t)}$ erases $K_{U_i}^{(t-1)}$ and any temporary storage after obtaining $K_{U_i}^{(t)}$. If the rekeying event is incurred by a Leave event, every $U_i \in \Delta^{(t)}$ erases $K_{U_i}^{(t-1)}$ and any temporary storage after obtaining $K_{U_i}^{(t)}$, and every *honest* leaving group member $U_j \in \Delta'$ erases $K_{U_j}^{(t-1)}$ (although a *corrupt* one does not have to follow this protocol).

We require that any group communication scheme satisfy the following `correctness` requirement: for any $t = 1, 2, \ldots$ and $\forall\, U \in \Delta^{(t)}$, if $\mathsf{acc}_U^{(t)} = \text{TRUE}$, then $k_U^{(t)} = k^{(t)}$ and $K_U^{(t)} \subset K_{\mathcal{GC}}^{(t)}$.

3.2. *Security definitions*

We consider an adversary that has complete control over all the communications in the network. To simplify the definition, we assume that the group controller is never compromised; this is not necessarily a restriction because the adversary could have compromised all the group members (and thus have obtained the secrets the group controller holds).

An adversary's interaction with principals in the network is modeled by allowing it to have access to (some of) the following oracles:

- $\mathcal{O}_{\mathit{Send}}(U, t, M, \mathit{action})$: Send a message M to $U \in \{\mathcal{GC}\} \cup \mathbb{ID}$ at time $t \geqslant 0$, and output its reply, where *action* $\in \{\mathsf{Setup}, \mathsf{Join}, \mathsf{Leave}, \mathsf{Rekey}\}$ meaning that U will execute according to the corresponding protocol, and M specifies the needed information for executing the protocol. Of course, the query of type Setup is only made at time $t = 0$.
These oracle accesses are meant to capture that the adversary can observe the reactions of the non-corrupt participants (e.g., the incurred message exchanges). For example, the adversary can let some honest (i.e., non-corrupt) users join or leave the group in question.
- $\mathcal{O}_{\mathit{Reveal}}(U, t)$: Output the group key held by $U \in \Delta^{(t)}$ at time t, namely $k_U^{(t)}$.
- $\mathcal{O}_{\mathit{Corrupt}}(U, t)$: Output the keys held by $U \in \Delta^{(t)}$ at time t, namely $K_U^{(t)}$.
- $\mathcal{O}_{\mathit{Test}}(U, t)$: This oracle may be queried only once, at any time during the adversary's execution. A random bit b is generated: if $b = 1$ the adversary is given $k_U^{(t)}$ where $U \in \Delta^{(t)}$, and if $b = 0$ the adversary is given a random key of length $|k_U^{(t)}|$.

Now we define the *active outsider attack model* that is strictly more powerful than the *passive outsider attack model* that has been implicitly utilized in the literature.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 15

Definition 3.1 (active outsider attack model). In this model, the adversary $\mathcal{A}$ may have access to all the oracles specified above. In particular, an "outsider" $\mathcal{A} \notin \Delta^{(t)}$ is allowed to issue an $\mathcal{O}_{Reveal}(U, t)$ or $\mathcal{O}_{Corrupt}(U, t)$ query for some $U \in \Delta^{(t)}$.

Definition 3.2 (passive attack model). In this model, the adversary is only allowed to make $\mathcal{O}_{Send}(\cdot, \cdot, \cdot, \cdot)$ and $\mathcal{O}_{Test}(\cdot, \cdot)$ queries. In other words, the adversary is only allowed to join and leave the group (in an arbitrary fashion though).

In each of the two models, we define two security notions: backward-security and forward-security. This leads to four security notions: (1) forward-security in the active outsider attack model or simply `security`, (2) backward-security in the active outsider attack model or simply `strong-security`, (3) forward-security in the passive attack model, and (4) backward-security in the passive attack model.

Definition 3.3 (`security`). Intuitively, it means that $\mathcal{A}$ learns no information about a group key if (1) with respect to the corresponding rekeying event there is no corrupt legitimate member (this implicitly implies that all the members that were corrupted by $\mathcal{A}$ must have been revoked), and (2) no member is corrupted by $\mathcal{A}$ after the rekeying event. Formally, consider the following event Succ:

(1) The adversary can make arbitrary oracle queries at any time $t_1 < t$, except the following restrictions hold.
(2) The adversary queries the $\mathcal{O}_{Test}(U, t)$ oracle with $\mathsf{acc}_U^{(t)} = \text{TRUE}$, and correctly guesses the bit b used by the $\mathcal{O}_{Test}(U, t)$ oracle in answering this query.
(3) There is no $\mathcal{O}_{Reveal}(V, t)$ query for any $V \in \Delta^{(t)}$. (Otherwise, the group key is trivially compromised.)
(4) For every $\mathcal{O}_{Corrupt}(V, t_1)$ query where $t_1 < t$, there must have been an $\mathcal{O}_{Send}(\mathcal{GC}, t_2, V, \mathsf{Leave})$ query where $t_1 < t_2 \leqslant t$. This captures that the corrupt members must have been revoked before the rekeying message at time t.
(5) There is no $\mathcal{O}_{Corrupt}(V, t_3)$ query for any $t_3 \geqslant t$ and $V \in \Delta^{(t_3)}$.

The advantage of the adversary $\mathcal{A}$ in attacking the group communication scheme is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by $\mathcal{GC}$ and by $\mathcal{A}$. We say a scheme is `secure` if for all probabilistic polynomial-time adversary $\mathcal{A}$ it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Definition 3.4 (`strong-security`). Intuitively, it means that an adversary learns no information about a group key if, with respect to the rekeying event of interest there is no corrupt legitimate member (this implicitly implies that all the previously corrupt members have been revoked). Formally, consider the following event Succ:

(1)–(4) The same as in the definition of `security`.
(5) There is no $\mathcal{O}_{Corrupt}(V, t_3)$ query for $t_3 = t$ and $V \in \Delta^{(t_3)}$. (This does not rule out that there could be some $\mathcal{O}_{Corrupt}(V, t_3)$ query for $t_3 > t$.)

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 16

The advantage of the adversary $\mathcal{A}$ in attacking the group communication scheme is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by $\mathcal{GC}$ and by $\mathcal{A}$. We say a scheme is `strongly-secure` if for all probabilistic polynomial-time adversary $\mathcal{A}$ it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Definition 3.5 (forward-security in the passive attack model). Intuitively, it means that $\mathcal{A}$, which is not allowed to make any $\mathcal{O}_{Reveal}$ or $\mathcal{O}_{corrupt}$ query, learns no information about any group key after leaving the group. Formally, consider the following event Succ:

(1) The adversary arbitrarily queries the $\mathcal{O}_{Send}(\cdot, t_1, \cdot, \cdot)$ oracle for any $t_1 < t$. Moreover, the adversary itself can arbitrarily join or leave the group at time t_1, provided that the following restriction holds.
(2) The adversary queries the $\mathcal{O}_{Test}(U, t)$ oracle, where (1) $\mathsf{acc}_U^{(t)} = \text{TRUE}$ for an honest user U, and (2) $\mathcal{A} \notin \Delta^{(t)}$. Then, the adversary correctly guesses the bit b used by the $\mathcal{O}_{Test}(U, t)$ oracle in answering this query.

The advantage of the adversary $\mathcal{A}$ in attacking the group communication scheme is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by $\mathcal{GC}$ and by $\mathcal{A}$. We say a scheme is `secure` if for all probabilistic polynomial-time adversary $\mathcal{A}$ it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Definition 3.6 (backward-security in the passive attack model). Intuitively, it means that $\mathcal{A}$, which is not allowed to make any $\mathcal{O}_{Reveal}$ or $\mathcal{O}_{corrupt}$ query, learns no information about any group key before joining the group (again). Formally, consider the following event Succ:

(1) The adversary may arbitrarily query the $\mathcal{O}_{Send}(\cdot, t_1, \cdot, \cdot)$ oracle for any $t_1 < t$. Moreover, $\mathcal{A}$ can arbitrarily join or leave the group at time t_1, provided that the following restriction holds.
(2) The adversary queries the $\mathcal{O}_{Test}(U, t)$ oracle, where (1) $\mathsf{acc}_U^{(t)} = \text{TRUE}$ for an honest user U, and (2) $\mathcal{A} \notin \Delta^{(t)}$.
(3) The adversary queries the $\mathcal{O}_{Send}(\cdot, t_2, \cdot, \cdot)$ oracle for any $t_2 > t$. Moreover, $\mathcal{A}$ can arbitrarily join or leave the group at time t_2.
(4) The adversary correctly guesses the bit b used by the $\mathcal{O}_{Test}(U, t)$ oracle in answering this query.

The advantage of the adversary $\mathcal{A}$ in attacking the group communication scheme is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by $\mathcal{GC}$ and by $\mathcal{A}$. We say a scheme is `secure` if for all probabilistic polynomial-time adversary $\mathcal{A}$ it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

It is trivial to see that `strong-security` implies *backward-security in the passive model*, and that `security` implies *forward-security in the passive attack model*.

3.3. Relationships between the security notions

We summarize the relationships between the security notions of stateful group communication schemes in Fig. 3, where $X \to Y$ means X is stronger than Y, $X \leftrightarrow Y$ means X is equivalent to Y, $X \not\to Y$ means X does not imply Y, and $X \overset{?}{\not\to} Y$ means it is unclear where X does not imply Y. Below we elaborate on the non-trivial relationships showed in Fig. 3.

Proposition 3.1. *If a stateful group communication scheme is* `strongly-secure`*, then it is also* `secure`*.*

Proof. This is almost immediate because, on one hand, the definition of `strong-security` ensures the secrecy of $k^{(t)}$ even if $\mathcal{A}$ corrupts some $U \in \Delta^{(t_3)}$ where $t_3 > t$, and on the other hand, the definition of `security` ensures the secrecy of $k^{(t)}$ only if $\mathcal{A}$ does not corrupt any $U \in \Delta^{(t_3)}$ for any $t_3 > t$. □

Proposition 3.2. *A stateful group communication scheme that is* `secure` *is not necessarily* `strong-secure`*.*

Proof. The fact that `security` does not imply `strong-security` is implied by Theorem 5.1, which states that LKH is `secure`, and that LKH is insecure against an active outside attacker (cf. the attack scenario in Section 1.1). □

The above proposition implies that for a stateful group communication scheme, one only needs to show that it is `strongly-secure`.

Proposition 3.3. *A stateful group communication scheme is backward-secure in the passive attack model iff it is forward-secure in the passive attack model.*

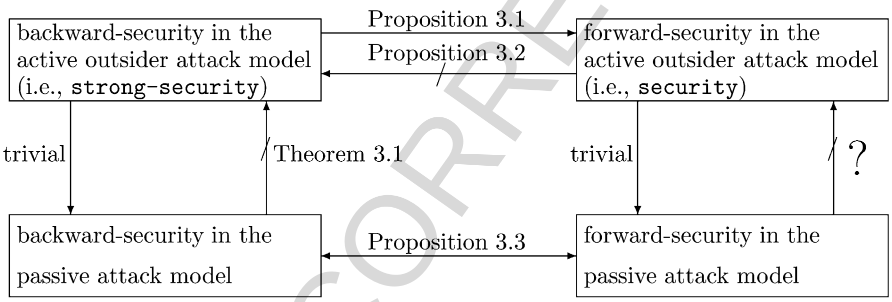

Fig. 3. The relationships between the security notions in stateful group communication schemes.

Proof. First we show that a group communication scheme that is not forward-secure in the passive attack model is also not backward-secure in the passive attack model. Suppose $\mathcal{A}$ first joins the group at time t_1 and then leaves the group at time t_2 where $t_1 < t_2$. Since the scheme is not forward-secure in the passive attack model, $\mathcal{A}$ can distinguish $k^{(t_3)}$ from a random string for some $t_3 > t_2$ with a non-negligible probability. Now suppose $\mathcal{A}$ re-joins the group at time t_4 for some $t_4 > t_3$. Then, with respect to this re-joining event, $\mathcal{A}$ can distinguish $k^{(t_3)}$ from a random string with a non-negligible probability. Since $\mathcal{A}$ did not make any $\mathcal{O}_{Reveal}$ or $\mathcal{O}_{corrupt}$ query, the scheme is not backward-secure in the passive attack model.

Now we show that a group communication scheme that is not backward-secure in the passive attack model is also not forward-secure in the passive attack model. Suppose $\mathcal{A}$ first joins the group at time t_1, leaves the group at time t_2, and re-joins the group at time t_3, where $t_1 < t_2 < t_3$. Since the scheme is not backward-secure in the passive attack model, $\mathcal{A}$ can distinguish $k^{(t)}$ from a random string for some $t_2 \leqslant t < t_3$ with a non-negligible probability. This also means that, with respect to the leaving event at time t_2, $\mathcal{A}$ can distinguish $k^{(t)}$ from a random string for some $t \geqslant t_2$ with a non-negligible probability. Since $\mathcal{A}$ does not make any $\mathcal{O}_{Reveal}$ or $\mathcal{O}_{corrupt}$ query, the scheme is not forward-secure in the passive attack model. □

We do not know whether forward-security in the passive attack mode also implies forward-security in the active outsider attack model. The relationship may seem trivial at a first glance, since all the corrupt members are revoked before the "challenge" session, and the adversary is not allowed to corrupt any member after the "challenge" session. Although it can indeed be shown that the implication holds, provided that the adversary is *static* (meaning that the adversary decides which principals in $\mathbb{ID}$ it will corrupt at system initialization), in the more interesting case that the adversary is *adaptive*, we do not know how to prove it.

Theorem 3.1. *There exists a group communication scheme that is backward-secure in the passive attack model but not* `strongly-secure` (*i.e., backward-secure in the active outsider attack model*).

Proof. Theorem 5.1 shows that LKH is `secure` (i.e., forward-secure in the active outsider attack model), which trivially means that it is also forward-secure in the passive attack model. Then, Proposition 3.3 shows that it is also backward-secure in the passive attack model.

On the other hand, the attack scenario shown in Section 1.1 states that LKH is not backward-secure in the active outsider attack model. □

4. A compiler for stateful group communication schemes

Suppose $\{f_k\}$ is a secure pseudorandom function family. Now we present a compiler that transforms a `secure` group communication scheme, SGC =

(Setup, Join, Leave, Rekey), into a `strongly-secure` one, SSGC = (Setup*, Join*, Leave*, Rekey*). The compiler applies to the subclass of stateful group communication schemes where the different keys belong to $K_{\mathcal{GC}}^{(t)}$ are computationally independent of each other, where $t = 0, 1, 2, \ldots$. In what follows "a key k needs to be changed" means that it should be substituted with a random key that is information-theoretically independent of k.

The key idea behind the compiler is to update the keys, which are possibly used to encrypt the new keys that need to be securely sent to the legitimate users, at each join, leave, or rekey event via an appropriate family of pseudorandom functions. As a result, compromise of a current key does not allow the adversary to recover the corresponding past keys.

Setup*: This is the same as SGC.Setup.

Join*: This algorithm is executed by $\mathcal{GC}$ at time, say, t. Let K be the set of keys that need to be changed (including the group key $k^{(t-1)}$), K^* be the set of common key(s) shared between the $\mathcal{GC}$ and the joining user(s), and K^{**} be the new keys (including the new group key $k^{(t)}$) that are used to replace the keys in K.

1. Execute SGC.Join except for the following: (1) for every $k_i \in (K_{\mathcal{GC}}^{(t-1)} \setminus \{k^{(t-1)}\}) \cup K^*$, let $f_{k_i}(0)$ play the role of k_i in SGC.Join; (2) for every $k_i \in K^{**} \setminus \{k^{(t)}\}$ that is used as an encryption key in SGC.Join, let $f_{k_i}(0)$ play the role of k_i.
2. Every individual key $k_i \in (K_{\mathcal{GC}}^{(t-1)} \setminus K) \cup K^* \cup (K^{**} \setminus \{k^{(t)}\})$ is replaced by $f_{k_i}(1)$.

Leave*: This algorithm is executed by $\mathcal{GC}$ at time, say, t. Let K be the set of keys that need to be changed (including the group key $k^{(t-1)}$) or eliminated, and K^{**} be the new keys (including the new group key $k^{(t)}$) that are used to replace (possibly a subset of) the keys in K.

1. Execute SGC.Leave except for the following: (1) for every $k_i \in K_{\mathcal{GC}}^{(t-1)} \setminus K$, let $f_{k_i}(0)$ play the role of k_i in SGC.Leave; (2) for every $k_i \in K^{**} \setminus \{k^{(t)}\}$ that is used as an encryption key in SGC.Leave, let $f_{k_i}(0)$ play the role of k_i.
2. Every individual key $k_i \in (K_{\mathcal{GC}}^{(t-1)} \setminus K) \cup (K^{**} \setminus \{k^{(t)}\})$ is replaced by $f_{k_i}(1)$.

Rekey*: There are two cases.

- The rekeying event is incurred by a Leave event at time t. In this case, every honest leaving user should erase all the secrets as in SGC.Rekey, and every remaining user, $V \in \Delta^{(t)}$, executes the following. Denote by $K'_V \subseteq K_V^{(t-1)}$ the subset of keys that need to be changed to a set of new keys K''_V. (We notice that both K'_V and K''_V can be derived by V after receiving the rekeying message and that $k^{(t)} \in K''_V$.) First, V executes

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 20

SGC.Rekey except for letting $f_{k_i}(0)$ play the role of k_i under the circumstance that $k_i \in K_V^{(t-1)} \setminus \{k^{(t-1)}\}$ or $k_i \in K''_V \setminus \{k^{(t)}\}$ is used as an encryption key, and updates every $k_i \in (K_V^{(t-1)} \setminus K'_V) \cup (K''_V \setminus \{k^{(t)}\})$ as $f_{k_i}(1)$. Second, V erases the outdated keys (except $K_V^{(t)}$) as in SGC.Rekey.

- The rekeying event is incurred by a Join event at time t. We notice that user $V \in \Delta^{(t)}$ holds a set of keys $K_V^{(t-1)}$ (in the case of V being a joining user, $K_V^{(t-1)}$ consists of the only common key between $\mathcal{GC}$ and V), of which a subset K'_V of keys (which may be empty) are to be changed to a set of new keys K''_V. (We notice that both K'_V and K''_V can be derived by V after receiving the rekeying message and that $k^{(t)} \in K''_V$.) First, V executes SGC.Rekey except for letting $f_{k_i}(0)$ play the role of k_i under the circumstance that $k_i \in K_V^{(t-1)} \setminus \{k^{t-1}\}$ or $k_i \in K''_V \setminus \{k^{(t)}\}$ is used as an encryption key, and updates every $k_i \in (K_V^{(t-1)} \setminus K'_V) \cup (K''_V \setminus \{k^{(t)}\})$ as $f_{k_i}(1)$. Second, $V \in \Delta^{(t)}$ erases the outdated keys (other than $K_V^{(t)}$) as in SGC.Rekey.

4.1. Analysis

First we analyze the complexity of SSGC.

- It does not introduce any extra communication complexity over SGC; this is important in many applications such as MANETs and sensor networks. (In Section 4.2 we further reduce the communication complexity.)
- It does not introduce any extra storage complexity over SGC, provided that the temporary storage for the keys such as $f_{k_i}(0)$ is insignificant. This is at least true for most applications including MANETs and sensor networks.
- The only extra complexity of SSGC over SGC is the evaluation of the pseudorandom functions. Specifically, the server needs to conduct $O(\max\{|K_{\mathcal{GC}}^{(t-1)}|, |K_{\mathcal{GC}}^{(t)}|\})$ pseudorandom function evaluation operations; a user V needs to conduct $O(\max\{|K_U^{(t-1)}|, |K_U^{(t)}|\})$ pseudorandom function operations. We notice that typically $|K_U^{(t)}| = O(\log(|K_{\mathcal{GC}}^{(t)}|))$ (e.g., [28]). This should be insignificant for most applications including MANETs and sensor networks.

Now we prove that SSGC is indeed `strongly-secure`. The intuition that SSGC is `strongly-secure` (and thus defeats the attacks presented in the Introduction) is due to the following fact: compromise of all of the current keys held by a user does not necessarily allow the adversary to recover any of the past keys, which may have been used to secure the transmission of other keys.

Theorem 4.1. *Assume $\{f_k\}$ is a secure pseudorandom function family (as specified in* Section 2*). If* SGC *is* `secure`*, then* SSGC *is* `strongly-secure`.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 21

Proof (sketch). We show that if SSGC is not `strongly-secure`, then SGC is not `secure`. Note that the key difference between the two security notions is whether the adversary is allowed to corrupt a legitimate user after the rekeying event of interest. Note also that after a rekeying event in SSGC all the new keys are either information-theoretically or computationally independent of each other.

First, consider a mental scheme that is the same as in SSGC except that the keys of the form $f_{k_i}(0)$ are always substituted with freshly and independently chosen random keys, where k_i is not held by any corrupt user. We claim that this mental scheme achieves `strong-security`; otherwise, there is an efficient algorithm to break the `security` of SGC. To see this, we construct a simulator that has access to a challenge SGC environment. Since the number of rekeying events in SSGC is polynomially-bounded, the simulator has an inverse polynomial probability in successfully guessing the rekeying event of interest – the event corresponding to the $\mathcal{O}_{Test}(\cdot,\cdot)$ query.

1. The simulator interacts with the adversary as in SSGC; this can be done because the simulator has complete control over the keys utilized in the SSGC. We notice that the simulator can answer any queries, including $\mathcal{O}_{reveal}$ and $\mathcal{O}_{corrupt}$.
2. When the simulated SSGC execution reaches the point of interest, the simulator asks the challenge SGC environment to establish an instance of SGC with the same set of users. The establishment of the instance is based on the rekeying event incurred by the adversary in SSGC, so that the legitimate users hold the corresponding keys as in SGC.Setup. This substitution can get through because the definition of `strong-security` ensures that the adversary in SSGC does not corrupt any legitimate user during the rekeying event.
3. At the next rekeying event, the simulator can continue its execution of the SSGC because it can utilize independent secrets that are freshly chosen by itself. We notice that the simulator can answer any queries, including $\mathcal{O}_{reveal}$ and $\mathcal{O}_{corrupt}$, as it can simulate the SSGC environment in any future rekeying events.

Second, it is clear that the difference between SSGC and the aforementioned mental scheme is that the keys utilized in the rekeying events are either series of keys in the forms of $f_{k_i}(1)$ where the k_i's are secret from the adversary, or freshly and independently chosen at random. We claim that the two cases are indistinguishable as long as the pseudorandom function family is secure. To see this, we notice that no adversary can, with a non-negligible probability, distinguish a single key-chain of a fixed key identity, namely $f_{k_i}(1)$, $f_{f_{k_i}(1)}(1)$, $f_{f_{f_{k_i}(1)}(1)}(1), \ldots$ where k_i is secret from the adversary, from a sequence of random secrets. Otherwise, we can construct an algorithm to distinguish a pseudorandom function from a random one with a non-negligible probability (because the number of rekeying events is polynomially-bounded). Conditioned on the fact that the number of keys is polynomially-bounded,

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 22

we conclude that the keys derived from pseudorandom functions are indistinguishable from the keys that are freshly and independently chosen; otherwise, a standard hybrid argument shows that there exists an algorithm that is able to distinguish a pseudorandom function from a random one (because there are at most a polynomial number of key chains). □

4.2. Performance optimization

In this section we show how to reduce the communication complexity in the SSGC; this might be very useful in applications such as MANETs and sensor networks. Suppose a Join event occurs at time t. The key observation includes:

1. In Join* of SSGC we could simply let the server sends the updated keys to the joining user U. We notice that, before U receiving the rekeying message from the server, $K_U^{(t-1)}$ consists of a single key, denoted by k^*, that is also known to the server. After sending the rekeying message, the server update $K_{\mathcal{GC}}^{(t-1)} = \{k_i\}$ to $K_{\mathcal{GC}}^{(t)} = \{f_{k_i}(1)\} \cup \{f_{k^*}(1)\}$.
2. When the joining user U executes Rekey* corresponding to the Join* (i.e., after receiving the rekeying message), it lets $f_{k^*}(0)$ plays the role of k^*. Then, U updates k^* to $f_{k^*}(1)$ while keeping intact the other keys received from the server.
3. When an existing user $V \in \Delta^{(t-1)}$ executes Rekey* corresponding to the Join*, it simply updates every $k_i \in K_V^{(t-1)}$ (including the group key) to $f_{k_i}(1)$.
4. The encryption of group communications is based on new group key $k^{(t)} = f_{k^{(t-1)}}(1)$.

We notice that the idea of substituting k_i via a certain function was pointed out in [8,26] with respect to the specific scheme of [28]. Here we show that it can actually be extended to accommodate the class of group communication schemes discussed in this paper. This justifies why we treat it as a possible feature of the compiler, which we call the *optimized compiler*.

Theorem 4.2. *Assume $\{f_k\}$ is a secure pseudorandom function family, and* SGC *is* `secure`. *If* SGC *does not adopt the afore-discussed performance optimization (otherwise, the optimized compiler does not gain anything over the original compiler), then the scheme output by the optimized compiler is also* `strongly-secure`.

The proof is similar to the proof of Theorem 4.1, and thus omitted.

5. A concrete `Strongly-Secure` stateful group communication scheme

In the last section we presented a compiler that can transform a certain `secure` stateful group communication scheme into a `strongly-secure` one. In

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 23

this section we present a concrete `strongly-secure` stateful group communication scheme, which is obtained by applying the compiler to LKH [28] that is shown to be `secure` in Section 5.3. First we briefly review LKH.

5.1. The model of LKH

The model of LKH is best known as a *key tree*, which outperforms the others (e.g., star key graph, or general key graph which actually leads to a certain NP-hard problem as we always need to minimize the communication complexity). A key tree T can be seen as a special class of directed acyclic graph with two types of nodes: u*-nodes* representing users and k*-nodes* representing keys. Each u-node is a leaf that has one outgoing edge but no incoming edge, and each k-node is an inner node that has one or more incoming edges. Moreover, there is a k-node (i.e., the root) that has incoming edges but no outgoing edge. In other words, the edges go from leaves towards the root.

Let U be a finite and nonempty set of users and K be a finite and nonempty set of keys. We are interested in a relation, $R \subseteq U \times K$, that can be specified by a key tree T as follows:

- There is a one-to-one correspondence between U and the set of u-nodes in T.
- There is a one-to-one correspondence between K and the set of k-nodes in T.
- $(u, k) \in R$ if and only if there is a directed path in T from the u-node that corresponds to a user $u \in U$ to the k-node that corresponds to a key $k \in K$.

This means that the group key is at the root of the tree, which is shared by all the users in U. Since a key tree can be specified by two parameters – the height h of the tree is the length (in number of edges) of the longest directed path in the tree, and the degree d of the tree is the maximum number of incoming edges of a node in the tree – each user in U has at most h keys.

In order to clarify the presentation, we define two functions, $\text{KEYSET}: U \rightarrow K$ and $\text{USERSET}: K \rightarrow U$, as follows:

$$\text{KEYSET}(u) = \{k|(u,k) \in R\},$$
$$\text{USERSET}(k) = \{u|(u,k) \in R\}.$$

Intuitively, $\text{KEYSET}(u)$ is the set of keys held by user $u \in U$, and $\text{USERSET}(k)$ is the set of users that hold key $k \in K$. Moreover, it is natural to generalize the definitions of $\text{KEYSET}(u \in U)$ to $\text{KEYSET}(U' \subseteq U) = \bigcup_{u \in U'} \text{KEYSET}(u)$, and of $\text{USERSET}(k \in K)$ to $\text{USERSET}(K' \subseteq K) = \bigcup_{k \in K'} \text{USERSET}(k)$.

5.2. A `Strongly-Secure` stateful group communication scheme

The new scheme is obtained by applying the compiler described in Section 4 to LKH based on the so-called *group-oriented* rekeying strategy, which is reviewed in

Appendix A for completeness. (LKH can be based on the less efficient *key-oriented* and *user-oriented* strategies [28]. Nevertheless, it should be straightforward to adapt our scheme to these rekeying strategies.) The scheme consists of four protocols, namely $\mathsf{SSGC} = (\mathsf{Setup}^*, \mathsf{Join}^*, \mathsf{Leave}^*, \mathsf{Rekey}^*)$.

Setup^*: The key server generates a key k_i for each k-node. After the initialization, each user (corresponding to a u-node) holds the keys corresponding to the path from its parent k-node to the root.
For example, if the initial system configuration is like in Fig. 1(a), then user u_5 holds keys, k_5, k_{456}, k_{1-8}, where k_{1-8} is the group key.

Join^*: After granting a join request from user u, the key server s creates a new u-node for user u and a new k-node for its individual key k_u. Then, server s finds an existing k-node (called the *joining point* for this join request) in the key tree and attaches the k-node k_u to the joining point as its child. As a consequence, the keys corresponding to the path – starting at the joining point and ending at the root – need to be changed. The algorithm is specified in Fig. 4, whose basic idea can be summarized as follows:

1. For each k-node x whose key needs to be changed, say, from $\bar{\mathsf{k}}_i$ to freshly chosen $\hat{\mathsf{k}}_i$, the server constructs two rekeying messages. The first rekeying message is the encryption of new key $\hat{\mathsf{k}}_i$ with $f_{\bar{\mathsf{k}}_i}(0)$, where $\bar{\mathsf{k}}_i$ is a non-root key that needs to be changed, and is sent to $\text{USERSET}(\bar{\mathsf{k}}_i)$, namely the set of users that share $\bar{\mathsf{k}}_i$. The second rekeying message contains the encryption of the new key $\hat{\mathsf{k}}_i$ with the individual key of the joining user, and is sent to the joining user. Moreover, these rekeying messages are appropriately grouped together.

Join protocol for group-oriented rekeying: // suppose user u joins the group
 server s generates a new key k_u for user u
 server s finds a joining point x_j
 server s attaches k_u to x_j
 let x_0 be the root
 denote by x_{i-1} the parent of x_i for $1 \leqslant i \leqslant j$
 $\bar{\mathsf{k}}_{j+1} \leftarrow k_u$
 let $\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_j$ be the current keys of $x_0, \ldots, x_j$, respectively
 server s generates fresh keys $\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j$ // new keys of $x_0, \ldots, x_j$
 $s \rightarrow \text{USERSET}(\bar{\mathsf{k}}_0): \{\hat{\mathsf{k}}_0\}_{\bar{\mathsf{k}}_0}, \{\hat{\mathsf{k}}_1\}_{f_{\bar{\mathsf{k}}_1}(0)}, \ldots, \{\hat{\mathsf{k}}_j\}_{f_{\bar{\mathsf{k}}_j}(0)}$
 $s \rightarrow \{u\}: \{\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j\}_{f_{\bar{\mathsf{k}}_{j+1}}(0)}$
 FOR all $\bar{\mathsf{k}} \in (\text{KEYSET}(\text{USERSET}(\bar{\mathsf{k}}_0)) \setminus \{\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_j\}) \cup \{\bar{\mathsf{k}}_{j+1}\} \cup \{\hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j\}$
  $\bar{\mathsf{k}} \leftarrow f_{\bar{\mathsf{k}}}(1)$

Fig. 4. Join-incurred group-oriented rekeying.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 25

2. Any other key $\bar{\mathsf{k}}_j$ that needs not to be changed is replaced by $f_{\bar{\mathsf{k}}_j}(1)$.

For example, if u_9 joins the group configured as in Fig. 1(a), then u_9 is granted to join at joining point of k-node k_{78}. Then, the group key is changed from k_{1-8} to k_{1-9}, and k_{78} is replaced with a new k_{789}. The rekeying messages sent to the users are:

$$\begin{aligned} s &\to \{u_1, \ldots, u_8\} : \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{f_{k_{78}}(0)}, \\ s &\to \{u_9\} \qquad\quad : \{k_{1-9}, k_{789}\}_{f_{k_9}(0)}. \end{aligned}$$

Finally, k_i is substituted with $f_{k_i}(1)$ for $i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 123, 456, 789\}$. The attack presented in the Introduction is blocked because, for example, compromise of $f_{k_9}(1)$ does not lead to the exposure of $f_{k_9}(0)$, where k_9 is not known to the adversary (because it has been securely erased). As a result, the past group key k_{1-9} cannot be recovered by the adversary.

Leave*: After granting a leave request from user u, the key server s updates the key tree by deleting the u-node for user u and the k-node for its individual key from the key tree. The parent of the k-node corresponding to the user's individual key is called the *leaving point*. As a consequence, the keys corresponding to the path – starting at the leaving point and ending at the root – need to be changed. The algorithm is specified in Fig. 5, whose basic idea can be summarized as follows:

1. For each k-node x whose key needs to be changed, say, from $\bar{\mathsf{k}}_i$ to freshly chosen $\hat{\mathsf{k}}_i$, the server constructs a rekeying message that is the encryption of $\hat{\mathsf{k}}_i$ with the keys of x's children in the new key tree. Note that "the keys of x's children in the new key tree" are either certain new keys that need to be distributed, or some current keys that need not to be changed (although they will be appropriately updated).
2. Any other key $\bar{\mathsf{k}}_j$ that needs not to be changed is replaced by $f_{\bar{\mathsf{k}}_j}(1)$.

For example, if u_8 leave the group as configured in Fig. 1(b), the leaving point is the k-node k_{789}. Then, the group key is changed from k_{1-9} to $k_{1-7,9}$, and the key of leaving point is changed from k_{789} to $k_{7,9}$ in Fig. 1(c). The rekeying message sent to the users is:

$$\begin{aligned} s \to \{u_1, \ldots, u_7, u_9\} &: \{k_{1-7,9}\}_{f_{k_{123}}(0)}, \{k_{1-7,9}\}_{f_{k_{456}}(0)}, \{k_{1-7,9}\}_{f_{k_{7,9}}(0)}, \\ &\quad \{k_{7,9}\}_{f_{k_7}(0)}, \{k_{7,9}\}_{f_{k_9}(0)}. \end{aligned}$$

Finally, k_i is updated to $f_{k_i}(1)$ for $i \in \{1, 2, 3, 4, 5, 6, 7, 9, 123, 456\}$, and $k_{7,9}$ is updated to $f_{k_{7,9}}(1)$. The attack presented in the Introduction is blocked because, for example, compromise of $f_{k_9}(1)$ does not lead to the exposure of $f_{k_9}(0)$, where k_9 is not known to the adversary (because it has been securely erased). As a result, $k_{7,9}$, and thus the past group key $k_{1-7,9}$ cannot be recovered by the adversary.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 26

Leave protocol for group-oriented rekeying: // suppose u leaves the group
  let x_{j+1} be the deleted k-node for k_u
  $\bar{\mathsf{k}}_{j+1} \leftarrow k_u$
  server s finds the leaving point x_j (parent of k_u)
  server s removes $\bar{\mathsf{k}}_{j+1}$ from the key tree
  let x_0 be the root
  denote by x_{i-1} the parent of x_i where $1 \leqslant i \leqslant j$
  let $\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_j$ be the keys of $x_0, x_1, \ldots, x_j$ // they need to be changed
  server s generates fresh keys $\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j$ as the new keys of $x_0, x_1, \ldots, x_j$
  FOR $i = 0$ TO $j-1$
    let $\bar{\mathsf{k}}_{i_1}, \ldots, \bar{\mathsf{k}}_{i_{z_i}}$ be the keys at the children of x_i in the new key tree
      where $\bar{\mathsf{k}}_{i_a}$ is to be changed to $\hat{\mathsf{k}}_{i+1}$ for some $a \in \{1, \ldots, z_i\}$
    $L_i \leftarrow (\{\hat{\mathsf{k}}_i\}_{f_{\bar{\mathsf{k}}_{i_1}}(0)}, \ldots, \{\hat{\mathsf{k}}_i\}_{f_{\bar{\mathsf{k}}_{i_{a-1}}}(0)}, \{\hat{\mathsf{k}}_i\}_{f_{\hat{\mathsf{k}}_{i+1}}(0)},$
      $\{\hat{\mathsf{k}}_i\}_{f_{\bar{\mathsf{k}}_{i_{a+1}}}(0)}, \ldots, \{\hat{\mathsf{k}}_i\}_{f_{\bar{\mathsf{k}}_{i_z}}(0)})$
  let $\bar{\mathsf{k}}_{j_1}, \ldots, \bar{\mathsf{k}}_{j_{z_j}}$ be the keys at the children of x_j in the new key tree
  $L_j \leftarrow (\{\hat{\mathsf{k}}_j\}_{f_{\bar{\mathsf{k}}_{j_1}}(0)}, \ldots, \{\hat{\mathsf{k}}_j\}_{f_{\bar{\mathsf{k}}_{j_z}}(0)})$
  $s \rightarrow \mathrm{USERSET}(\bar{\mathsf{k}}_0) \setminus \{u\} : (L_0, \ldots, L_j)$
  FOR all $\bar{\mathsf{k}} \in (\mathrm{KEYSET}(\mathrm{USERSET}(\bar{\mathsf{k}}_0)) \setminus \{\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_{j+1}\}) \cup \{\hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j\}$
    $\bar{\mathsf{k}} \leftarrow f_{\bar{\mathsf{k}}}(1)$

Fig. 5. Leave-incurred group-oriented rekeying.

Rekey*: If the rekeying event is incurred by a join event, a legitimate user (i.e., an existing one or a joining one) obtains a subset Θ' of $\Theta = \{\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j\}$, and updates each $\hat{\mathsf{k}} \in \Theta' \setminus \{\hat{\mathsf{k}}_0\}$ to $f_{\hat{\mathsf{k}}}(1)$. If the rekeying event is incurred by a leave event, a legitimate user (i.e., one remaining in the group) obtains a subset Θ' of $\Theta = \{\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j\}$, and updates each $\hat{\mathsf{k}} \in \Theta' \setminus \{\hat{\mathsf{k}}_0\}$ to $f_{\hat{\mathsf{k}}}(1)$. In any case, a legitimate user u updates each $k_i \in \mathrm{KEYSET}(u)$ to $f_{k_i}(1)$, as long as k_i is not changed to any key belonging to Θ, and erases the outdated keys.
For example, corresponding to the event that u_9 joins the group as shown in Fig. 1(a), u_1 obtains k_{1-9}, updates k_{123} to $f_{k_{123}}(1)$, and updates k_1 to $f_{k_1}(1)$. Whereas, u_9 obtains k_{1-9} as well as k_{789}, updates k_{789} to $f_{k_{789}}(1)$, and updates k_9 to $f_{k_9}(1)$. Corresponding to the event that u_8 leaves the group as shown in Fig. 1(b), u_1 obtains $k_{1-7,9}$, updates k_{123} to $f_{k_{123}}(1)$, and updates k_1 to $f_{k_1}(1)$. Whereas, u_9 obtains $k_{1-7,9}$ as well as $k_{7,9}$, updates $k_{7,9}$ to $f_{k_{7,9}}(1)$, and updates k_9 to $f_{k_9}(1)$.

5.3. Analysis

Theorem 5.1. *Assume that the stand-alone encryptions utilized in LKH are based on a secure pseudorandom function family. Then, LKH is* `secure`*.*

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 27

Proof (sketch). Consider a mental scheme that is the same as LKH, except that the encryptions corresponding to the rekeying event of interest – the event corresponding to the $\mathcal{O}_{Test}(\cdot,\cdot)$ query – are based on random functions. This substitution can get through because the definition of `security` requires that there are no corrupt users. We claim that this mental scheme is `secure`; otherwise, a standard hybrid argument shows that the pseudorandom function family is broken because the number of encryptions is polynomially-bounded (which is further based on the fact that the size of the key-tree is polynomially-bounded). Conditioned on the fact that there are a polynomially-bounded number of rekeying events, we conclude that LKH is `secure`. □

As a corollary of Theorem 4.1 (which states that the compiler transforms a `secure` stateful group communication scheme to a `strongly-secure` one) and Theorem 5.1 (which states that LKH is indeed `secure`), we have:

Corollary 5.1. *The scheme presented in* Section 5.2 *is* `strongly-secure`.

5.4. Performance optimization

The improved scheme differs from SSGC only in Join* and Rekey*.

Improved Join*: This algorithm is specified in Fig. 6. It is the same as the Join* except the following: (1) instead of freshly choosing new keys for the k-nodes on the path starting at a joining point and ending at the root, we simply update every existing key k as $f_k(1)$, and (2) the new group key for encrypting actual group communications is $f_k(0)$, where k is the already updated key at the root.

Improved Rekey*: It is the same as Rekey* except that when the rekeying is incurred by a join event: every existing user v holding a key set KEYSET(v) needs to

Join protocol for group-oriented rekeying: // suppose user u joins the group
  server s generates a new key k_u for user u
  server s finds a joining point x_j
  server s attaches k_u to x_j
  let x_0 be the root
  denote by x_{i-1} the parent of x_i for $1 \leqslant i \leqslant j$
  let $\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_j$ be the current keys of $x_0, \ldots, x_j$, respectively
  $\bar{\mathsf{k}}_{j+1} \leftarrow k_u$
  $s \rightarrow \text{USERSET}(\bar{\mathsf{k}}_0)$: "key update"
  $s \rightarrow \{u\} : \{f_{\bar{\mathsf{k}}_0}(1), f_{\bar{\mathsf{k}}_1}(1), \ldots, f_{\bar{\mathsf{k}}_j}(1)\}_{f_{\bar{\mathsf{k}}_{j+1}}(0)}$
  FOR all $\bar{\mathsf{k}} \in \text{KEYSET}(\text{USERSET}(\bar{\mathsf{k}}_0)) \cup \{\bar{\mathsf{k}}_{j+1}\}$
    $\bar{\mathsf{k}} \leftarrow f_{\bar{\mathsf{k}}}(1)$

Fig. 6. Improved join-incurred group-oriented rekeying.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 28

update every $k \in \text{KEYSET}(v)$ to $f_k(1)$, whereas the joining user u only needs to update its common key k_u, which is established during the out-of-band approval of the join request, to $f_{k_u}(1)$. Note that the new group key for encrypting actual group communications is $f_k(0)$, where k is the already updated key at the root.

For example, if u_9 joins the group configured as in Fig. 1(a), then u_9 is granted to join at the joining point of k-node k_{78}. The key at the root is updated from k_{1-8} to $k_{1-9} = f_{k_{1-8}}(1)$ such that the new key for encrypting actual group communications is $f_{k_{1-9}}(0)$, and k_{78} is updated to $k_{789} = f_{k_{78}}(1)$. The sever sends the following messages:

$$\begin{aligned} &s \to \{u_1, \ldots, u_8\} &&: \text{“key update”} \\ &s \to \{u_9\} &&: \{k_{1-9}, k_{789}\}_{f_{k_9}(0)}. \end{aligned}$$

Every existing user u_i, $i \in \{1,2,3,4,5,6,7,8\}$, with key set K_i, updates every $k \in K_i$ as $f_k(1)$. For example, u_7 updates k_{1-8} to $k_{1-9} = f_{k_{1-8}}(1)$, updates k_{78} to $k_{789} = f_{k_{78}}(1)$, and updates k_7 to $k_7 = f_{k_7}(1)$. On the other hand, the joining user u_9 only needs to update k_9 to $k_9 = f_{k_9}(1)$, which means that it keeps $(k_{1-9}, k_{89}, k_9 = f_{k_9}(1))$.

As a corollary of Theorem 4.2 (which states that the optimized compiler in Section 4.2 transforms a `secure` stateful group communication scheme into a `strongly-secure` one) and Theorem 5.1 (which states that LKH is indeed `secure`), we have

Corollary 5.2. *The optimized scheme in* Section 5.4 *is* `strongly-secure`.

6. The case of stateless group communication schemes

Recall that we briefly reviewed the subset-cover framework [22] in Section 1.1. This section is organized as follows. In Section 6.1 we discuss the models and security definitions, including the notions of `strong-security` (i.e., backward-security in the active outsider attack model) and of `security` (i.e., forward-security in the passive attack model). In Section 6.2 we explore the relationships between the security notions. In Section 6.3 we present a compiler that can transform a *subclass* of `secure` stateless group communication schemes into `strongly-secure` ones, whose security is analyzed in Section 6.4. A concrete `strongly-secure` stateless group communication scheme, which is based on the *complete subtree method* [22], is presented in Section 6.5. Some practical issues are discussed in Section 6.6.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 29

6.1. Model and security of stateless group communication schemes

The subset-cover framework of [22] was briefly reviewed in Fig. 2. More specifically, let κ be a security parameter, $\mathcal{N}$ be the set of all users such that $|\mathcal{N}| = N$ is polynomially-bounded, and $\mathcal{R} \subset \mathcal{N}$ be a group of $|\mathcal{R}| = r$ users whose decryption privileges should be revoked. Let E_L be a symmetric key cryptosystem secure against an adaptive chosen-plaintext attack, and F_K be a symmetric key cryptosystem with a weaker security property called indistinguishability under a single chosen-plaintext attack in [22] (which is called "IND-P0-C0 security" in [13]).[3]

Recall that the goal of a stateless group communication scheme is to allow a center (or group controller, server, or sender) to transmit a message M to all users such that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ cannot decrypt it. Suppose $S_1, \ldots, S_w$ are a collection of subsets of users, where $S_j \subseteq \mathcal{N}$ for $1 \leqslant j \leqslant w$, and each S_j is assigned a long-lived key L_j such that each $u \in S_j$ should be able to deduce L_j from its secret information I_u. Given a revoked set $\mathcal{R}$, if one can partition $\mathcal{N} \setminus \mathcal{R}$ into (ideally disjoint) sets $S_{i_1}, \ldots, S_{i_m}$ such that $\mathcal{N} \setminus \mathcal{R} \subseteq \cup_{\ell=1}^{m} S_{i_\ell}$, then a message-encryption key K can be encrypted m times with $L_{i_1}, \ldots, L_{i_m}$, and each user $u \in \mathcal{N} \setminus \mathcal{R}$ can obtain K and thus M.

In what follows, by "$\mathcal{A}$ corrupts a user u" we mean that not only the internal state of u (including I_u) is given to $\mathcal{A}$, but also u will behave under $\mathcal{A}$'s control (i.e., Byzantine); by "u is revoked" we mean that u is not entitled to receive the message with respect to the specified session(s). For simplicity, we assume that a user, once corrupted, is always corrupt.

Stateless group communication schemes are indeed simpler than stateful ones because (1) both the joining and leaving operations are implicit – the rekeying messages may even be coupled with the payload, and (2) when a user (re-)joins a group, its long-term keys can indeed be reused. Therefore, the model of stateless group communication schemes can also be correspondingly simplified. In particular, we assume the center keeps an incremental counter for each broadcast messages so that encryptions may be simply denoted by $C^{(1)}, C^{(2)}, \ldots$ and the corresponding plaintext messages may be denoted by $M^{(1)}, M^{(2)}, \ldots$. One may think each $C^{(i)}$ corresponds to an "rekeying" event with revocation set $\mathcal{R}^{(i)}$ for $i = 1, 2, \ldots$. Note that $\mathcal{R}^{(i)} \neq \mathcal{R}^{(i+1)}$.

[3]Notice that [22] required that E_L be secure against chosen-ciphertext attacks, whereas we require it to be secure against chosen-plaintext attacks. The reason is that we need to we assume that the underlying communication channels are authenticated. While this naturally prevents chosen-ciphertext attacks, it also avoid another subtle attack, namely that a dishonest user could successfully cheat an honest user into accepting an impersonating message. The reason is simply due to the fact that E_L being secure against chosen-ciphertext attacks does not necessarily prevent this attack, because the dishonest user also knows the common secret key. This subtlety is well understood in the context of group communications (cf. [5]).

We assume that during the system initialization the center can communicate with each legitimate user through an *authenticated private* channel. In practice, the authenticated private channel can be implemented by a two-party authenticated key-exchange protocol, which should also ensure, as in the case of stateful group communication schemes, that certain relevant keys are securely erased after the initialization. Further, we assume that after the system initialization the center can communicate with a user through an *authenticated* channel.

In parallel to the case of stateful group communication schemes, we define two adversarial models for stateless group communication schemes: the *active outsider attack model* and the *passive outsider attack model*.

Definition 6.1 (active outsider attack model). With respect to a given ith broadcast message $C^{(i)}$, we say a user u is legitimate if $u \in \mathcal{N} \setminus \mathcal{R}^{(i)}$, and is illegitimate (or an outsider) otherwise. By "active outsider attack model" we mean the adversarial model in which an outsider $\mathcal{A}$ of the ith broadcast message, which may be called the "challenge" message, is allowed to corrupt legitimate users of $C^{(j)}$ for any $j > i$ (i.e., $u \in \mathcal{N} \setminus \mathcal{R}^{(j)}$).

Definition 6.2 (passive attack model). In this adversarial model, the adversary $\mathcal{A}$ is not allowed to corrupt any other legitimate member. In other words, the adversary is only allowed to decide when it is to be revoked (though in an arbitrary fashion). Formally, $\mathcal{A}$ cannot corrupt any $u \in \mathcal{N} \setminus \{\mathcal{A}\}$ for $i = 1, 2, \ldots.$

In each of the two models, we define two security notions: backward-security and forward-security. That is, we have four security notions: (1) forward-security in the active outsider attack model or simply `security` for short, (2) backward-security in the active outsider attack model or simply `strong-security` for short, (3) forward-security in the passive attack model, and (4) backward-security in the passive attack model.

Definition 6.3 (`security`; adapted from [22]). Consider an adversary $\mathcal{A}$ that gets to

1. Select adaptively $\mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(\ell_1)}$ of receivers, obtain I_u for all $u \in \mathcal{R}^{(i)}$ and see $C^{(1)}, C^{(2)}, \ldots, C^{(\ell_1)}$ for $i = 1, 2, \ldots, \ell_1$.
2. Choose a message M as the challenge plaintext and a set $\mathcal{R}$ of revoked users that must include all the ones it corrupted (but may contain more); i.e., $\cup_{i=1}^{\ell_1} \mathcal{R}^{(i)} \subseteq \mathcal{R}$. $\mathcal{A}$ then receives an encrypted message C with a revoked set $\mathcal{R}$, where C is the encryption of either M or a random message of the same length. We may call this the "challenge" message.
3. For $i = \ell_1 + 2, \ell_1 + 3, \ldots$, the following restrictions apply. (1) Even if $\mathcal{A} \in \mathcal{N}$, $\mathcal{A}$ can only decide whether $\mathcal{A} \in \mathcal{R}^{(i)}$. (2) Even if $\mathcal{R}^{(i)} \setminus \{\mathcal{A}\} \neq \emptyset$, $\mathcal{A}$ has no access to any I_u for $u \in \mathcal{R}^{(i)} \setminus \{\mathcal{A}\}$.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 31

Now $\mathcal{A}$ has to guess whether C corresponds to the encryption of the real message M or a random message. Denote by Succ the event that $\mathcal{A}$ makes the right guess. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by the center and by $\mathcal{A}$. We say that a stateless group communication scheme is `secure` (or forward-secure in the active outsider attack model) if, for any probabilistic polynomial-time $\mathcal{A}$ as above, it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Definition 6.4 (`strong-security`). Consider an adversary $\mathcal{A}$ that gets to

1. Select adaptively $\mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(\ell_1)}$ of receivers, obtain I_u for all $u \in \mathcal{R}^{(i)}$ and see $C^{(1)}, C^{(2)}, \ldots, C^{(\ell_1)}$ for $i = 1, 2, \ldots, \ell_1$.
2. Choose a message M as the challenge plaintext and a set $\mathcal{R}$ of revoked users that must include all the ones it corrupted (but may contain more); i.e., $\cup_{i=1}^{\ell_1} \mathcal{R}^{(i)} \subseteq \mathcal{R}$. $\mathcal{A}$ then receives an encrypted message C with a revoked set $\mathcal{R}$, where C is the encryption of either M or a random message of the same length. We may call this the "challenge" message.
3. Select adaptively $\mathcal{R}^{(\ell_1+2)}, \mathcal{R}^{(\ell_1+3)}, \ldots$ of receivers and obtain I_u for all $u \in \mathcal{R}^{(i)}$ for $i = \ell_1 + 2, \ell_1 + 3, \ldots$. Besides, $\mathcal{A}$ may select messages $M^{(\ell_1+2)}, M^{(\ell_1+3)}, \ldots$ and see the encryption of $C^{(\ell_1+2)}, C^{(\ell_1+3)}, \ldots$.

Now $\mathcal{A}$ has to guess whether C corresponds to the encryption of the real message M or a random message. Denote by Succ the event that $\mathcal{A}$ makes the right guess. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by the center and by $\mathcal{A}$. We say that a stateless group communication scheme is `strongly-secure` (or backward-secure in the active outsider attack model) if, for any probabilistic polynomial-time $\mathcal{A}$ as above, it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Definition 6.5 (forward-security in the passive attack model). Consider an adversary $\mathcal{A}$ that gets to

1. Select adaptively $\mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(\ell_1)}$ of receivers, and see $C^{(1)}, C^{(2)}, \ldots, C^{(\ell_1)}$ for $i = 1, 2, \ldots, \ell_1$. However, $\mathcal{A}$ does not have access to any I_u, where $u \in \mathcal{R}^{(i)} \setminus \{\mathcal{A}\}$ and $i = 1, 2, \ldots, \ell_1$.
2. Choose a message M as the challenge plaintext and a set $\mathcal{R}$ of revoked users that must include all the ones it corrupted (but may contain more); i.e., $\cup_{i=1}^{\ell_1} \mathcal{R}^{(i)} \subseteq \mathcal{R}$. $\mathcal{A}$ then receives an encrypted message C with a revoked set $\mathcal{R}$, where C is the encryption of either M or a random message of the same length. We may call this the "challenge" message.

Now $\mathcal{A}$ has to guess whether C corresponds to the encryption of the real message M or a random message. Denote by Succ the event that $\mathcal{A}$ makes the right guess.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by the center and by $\mathcal{A}$. We say that a stateless group communication scheme is `secure` (or forward-secure in the active outsider attack model) if, for any probabilistic polynomial-time $\mathcal{A}$ as above, it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Definition 6.6 (backward-security in the passive attack model). Consider an adversary $\mathcal{A}$ that gets to

1. Select adaptively $\mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(\ell_1)}$ of receivers, and see $C^{(1)}, C^{(2)}, \ldots, C^{(\ell_1)}$ for $i = 1, 2, \ldots, \ell_1$. However, $\mathcal{A}$ does not have access to any I_u, where $u \in \mathcal{R}^{(i)} \setminus \{\mathcal{A}\}$ and $i = 1, 2, \ldots, \ell_1$.
2. Choose a message M as the challenge plaintext and a set $\mathcal{R}$ of revoked users that must include all the ones it corrupted (but may contain more); i.e., $\cup_{i=1}^{\ell_1} \mathcal{R}^{(i)} \subseteq \mathcal{R}$. $\mathcal{A}$ then receives an encrypted message C with a revoked set $\mathcal{R}$, where C is the encryption of either M or a random message of the same length. We may call this the "challenge" message.
3. Select adaptively $\mathcal{R}^{(\ell_1+2)}, \mathcal{R}^{(\ell_1+3)}, \ldots$ of receivers, and possibly select messages $M^{(\ell_1+2)}, M^{(\ell_1+3)}, \ldots$ and see the encryption of $C^{(\ell_1+2)}, C^{(\ell_1+3)}, \ldots$. However, $\mathcal{A}$ does not have access to any I_u, where $u \in \mathcal{R}^{(i)} \setminus \{\mathcal{A}\}$ and $i = \ell_1 + 2, \ell_1 + 3, \ldots$.

Now $\mathcal{A}$ has to guess whether C corresponds to the encryption of the real message M or a random message. Denote by Succ the event that $\mathcal{A}$ makes the right guess. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\mathsf{Succ}] - 1|$, where $\Pr[\mathsf{Succ}]$ is the probability that the event Succ occurs, and the probability is taken over the coins used by the center and by $\mathcal{A}$. We say that a stateless group communication scheme is `strongly-secure` (or backward-secure in the active outsider attack model) if, for any probabilistic polynomial-time $\mathcal{A}$ as above, it holds that $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

It is trivial to see that `strong-security` implies *backward-security in the passive model*, and that `security` implies *forward-secure in the passive attack model*.

6.2. *Relationships between the security notions*

We summarize the relationships between the security notions of stateless group communication schemes in Fig. 7, where $X \rightarrow Y$ means X is stronger than Y, $X \leftrightarrow Y$ means X is equivalent to Y, $X \not\rightarrow Y$ means X does not imply Y, and $X \stackrel{?}{\rightarrow} Y$ means it is unclear where X implies Y. Below we elaborate on the non-trivial relationships showed in Fig. 7.

Proposition 6.1. *If a stateless group communication scheme is* `strongly-secure`*, then it is also* `secure`*.*

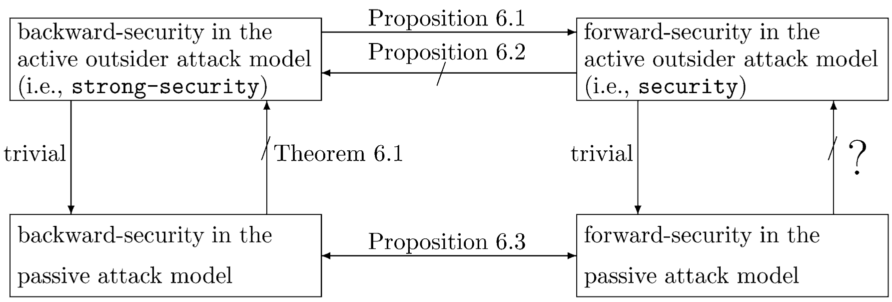

Fig. 7. The relationships between the security notions in stateless group communication schemes.

Proof. This is almost immediate because, on one hand, the definition of `strong-security` ensures the secrecy of the encrypted content of C even if $\mathcal{A}$ can have access to I_u for $u \in \mathcal{N} \setminus \mathcal{R}^{(i)}$ for $i = \ell_1 + 2, \ell_1 + 3, \ldots$, and on the other hand, the definition of `security` ensures the secrecy of the encrypted content of C only if $\mathcal{A}$ does not have access to any I_u such that $u \in \mathcal{N} \setminus \mathcal{R}^{(i)}$ and $i \in \{\ell_1+2, \ell_1+3, \ldots\}$. □

Proposition 6.2. *A stateless group communication scheme that is* `secure` *is not necessarily* `strong-secure`*.*

Proof. The fact that `security` does not imply `strong-security` is implied by (1) Theorem 6.3, which states that the complete subtree method of the subset-cover framework is `secure`, and (2) that the subset-cover framework is insecure against an active outsider attacker (cf. the attack scenario in Section 1.1). The key observation is indeed that the adversary's capability in the `strong-security` is strictly stronger. □

The above proposition implies that for a stateless group communication scheme, one only needs to show that it is `strongly-secure`.

Proposition 6.3. *A stateless group communication scheme is backward-secure in the passive attack model iff it is forward-secure in the passive attack model.*

Proof. First we show that a stateless group communication scheme that is not forward-secure in the passive attack model is also not backward-secure in the passive attack model. Suppose $\mathcal{A}$ is legitimate with respect to $C^{(i_1)}$, and illegitimate with respect to $C^{(i_2)}$ where $i_1 < i_2$. Since the scheme is not forward-secure in the passive attack model, $\mathcal{A}$ can derive some information about M corresponding to the $(\ell_1 + 1)$th broadcast C with a non-negligible probability, where $i_2 \leqslant \ell_1 + 1$. Now suppose $\mathcal{A}$ is legitimate with respect to $C^{(i_3)}$ where $\ell_1 + 1 < i_3$. Then, with respect to $C^{(i_3)}$, $\mathcal{A}$ can derive some information about a past encrypted message M with respect to the $(\ell_1 + 1)$th broadcast with a non-negligible probability. Since $\mathcal{A}$ does not corrupt any other legitimate user $u \in \mathcal{R}^j \setminus \{\mathcal{A}\}$ for $j = \ell_1 + 2, \ell_1 + 3, \ldots$, the scheme is not backward-secure in the passive attack model.

Second we show that a group communication scheme that is not backward-secure in the passive attack model is also not forward-secure in the passive attack model. Suppose $\mathcal{A} \in \mathcal{N} \setminus \mathcal{R}_{i_1}$, $\mathcal{A} \notin \mathcal{N} \setminus \mathcal{R}_{i_2}$, and $\mathcal{A} \in \mathcal{N} \setminus \mathcal{R}_{i_3}$, where $i_1 < i_2 < i_3$. Since the scheme is not backward-secure in the passive attack model, without loss of generality, $\mathcal{A}$ can derive some information about $M^{(i)}$ for some $i_2 \leqslant i < i_3$ with a non-negligible probability. This also means that, with respect to $C^{(i_2)}$, $\mathcal{A}$ can derive some information about a future message $M^{(i)}$ for some $i \geqslant i_2$. Since $\mathcal{A}$ does not corrupt any other legitimate users, the scheme is not forward-secure in the passive attack model. □

We do not know whether forward-security in the passive attack mode also implies forward-security in the active outsider attack model. The relationship may seem trivial at a first glance, since all the corrupt members are revoked before the "challenge" session, and the adversary is not allowed to corrupt any member after the "challenge" session. Although it can indeed be shown that the implication holds, provided that the adversary is *static* (meaning that the adversary decides which principals in $\mathcal{N}$ it will corrupt), in the more interesting case that the adversary is *adaptive*, we do not know how to prove it.

Theorem 6.1. *There exists a stateless group communication scheme that is "backward-secure in the passive attack model" but not* `strongly-secure` (*i.e., backward-secure in the active outsider attack model*).

Proof. Theorem 6.3 shows that the complete subtree revocation scheme in the subset-cover framework is `secure` (i.e., forward-secure in the active outsider attack model), which trivially means that it is also forward-secure in the passive attack model. Then, Proposition 6.3 shows that it is also backward-secure in the passive attack model.

On the other hand, the attack scenario showed in Section 1.1 states that the subset-cover framework is not backward-secure in the active outsider attack model. □

6.3. A compiler for stateless group communication schemes

Now we present a compiler that can transform a subclass of `secure` stateless group communication schemes falling into the subset-cover framework (called the input schemes) into `strongly-secure` ones. The subclass of stateless group communication schemes has the characteristics that the different keys belonging to $\{L_i\}_i \cup \{I_u\}_u$ are computationally independent of each other. Let $\{f_k\}$ be a pseudo-random function family. The compiler is specified in Fig. 8.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 35

Initialization: This is the same as in the input scheme.

Broadcasting: Given a set $\mathcal{R}$ (which may be empty at the first broadcasting after initialization), the center executes the following:

1. Choose a session encryption key K.
2. Find a partition of the users in $\mathcal{N} \setminus \mathcal{R}$ into disjoint subsets $S_{i_1}, \ldots, S_{i_m}$. Let $L_{i_1}, \ldots, L_{i_m}$ be the keys associated with the above subsets.
3. Encrypt K with keys $f_{L_{i_1}}(0), \ldots, f_{L_{i_m}}(0)$ and send the ciphertext

$$\langle [i_1, \ldots, i_m, E_{f_{L_{i_1}}(0)}(K), \ldots, E_{f_{L_{i_m}}(0)}(K)], F_K(M) \rangle.$$

1. Update L_i to $f_{L_i}(1)$ for all i if $\mathcal{R} \neq \emptyset$.

Decryption: A receiver u, upon receiving a broadcast message $\langle [i_1, \ldots, i_m, C_1, \ldots, C_m], C \rangle$, executes as follows.

1. Find i_j such that $u \in S_{i_j}$ (in the case $u \in \mathcal{R}$ the result is NULL).
2. Extract the corresponding key L_{i_j} from I_u.
3. Decrypt C_j using key $f_{L_{i_j}}(0)$ to obtain K.
4. Decrypt C using key K to obtain the message M.
5. Update L_i to $f_{L_i}(1)$ for all the i it holds if the broadcast is incurred by a revocation event (i.e., $m > 1$).

Fig. 8. The compiler for stateless group communication schemes.

6.4. Security analysis of the compiler

The key idea that the scheme resulting from the above compiler is not subject to the attack presented in the introduction is the following: compromise of a user at time t does not allow the adversary to recover keys corresponding to time $t_1 < t$. This is fulfilled by updating the keys using an appropriate family of pseudorandom functions.

Theorem 6.2. *Suppose the input scheme is* `secure`*, and* $\{f_k\}$ *is a secure pseudorandom function family, and the different keys belonging to* $\{L_i\}_i \cup \{I_u\}_u$ *are computationally independent of each other. Then, the above scheme is* `strongly-secure` *in the sense of* Definition 6.4.

Proof. Consider a mental game in which the system is initialized as in the input scheme. However, with respect to each broadcast operation, each incorrupt L_i is substituted with a pair of independently chosen random keys $\langle L_i^{(a,0)}, L_i^{(a,1)} \rangle$ such that $L_i^{(a,0)}$ is used to encrypt the message-encryption key K (if selected), where $a = 1, 2, \ldots$. We claim that this scheme is `secure`. This is because the keys that are used to encrypt the session key are freshly and independently chosen at random, which means that it is essentially a "short-lived" version of the input scheme. We also claim that this scheme is `strongly-secure`. This is because the keys that are used to encrypt the message-encryption key are freshly and independently chosen at random, which means that the secrets compromised after the "challenge" message are information-theoretically independent of the secrets used to encrypt the session key in the "challenge" message. Therefore, this scheme is `strongly-secure`.

Suppose the scheme output by the compiler (called the real-life scheme) is not `strongly-secure`. We observe that the difference between the above mental game and the real-life scheme is "how the incorrupt keys are evolved." Specifically, for $a > 0$, in the former case, the $\langle L_i^{(a,0)}, L_i^{(a,1)}\rangle$ are independently chosen at random; in the latter case, $\langle L_i^{(a,0)} = f_{f^{a-1}_{L_i}(1)}(0), L_i^{(a,1)} = f^a_{L_i}(1)\rangle$, where $f^0_X(\cdot) = X$, $f^1_X(\cdot) = f_X(\cdot)$, and $f^2_X(\cdot) = f_{f_X(\cdot)}(\cdot)$.

Now we consider the following experiment EXPT_j, where $0 \leqslant j \leqslant \ell$ and ℓ is the total number of revocation operations (which is polynomially bounded). The experiment is initialized as in the above mental game or as in the real-life scheme (both are the same at this stage). For any $0 \leqslant a \leqslant j$, any *incorrupt* $\langle L_i^{(a,0)}, L_i^{(a,1)}\rangle$ are independently chosen at random. For any $j < a \leqslant \ell$, any *incorrupt* $\langle L_i^{(a,0)}, L_i^{(a,1)}\rangle$ is defined as $\langle L_i^{(a,0)} = f_{f^{a-j-1}_{L_i^{(j,1)}}(1)}(0), L_i^{(a,1)} = f^{a-j}_{L_i^{(j,1)}}(1)\rangle$. The experiments can get through because the secrets (some of them are used for encrypting the message-encryption key) are (at least) computationally independent of each other. We observe that EXPT_0 corresponds to the real-life scheme, and EXPT_ℓ corresponds to the above mental scheme. Since we assumed that EXPT_0 is not `strongly-secure`, it holds that $\mathcal{A}$ has a non-negligible success probability ε_0 with respect to Definition 6.4. On the other hand, we already know that EXPT_ℓ is `strongly-secure`, which means that $\mathcal{A}$ has only a negligible success probability ε_ℓ. Since ℓ is polynomially bounded, there must exist $0 \leqslant j < \ell$ such that EXPT_j and EXPT_{j+1} are distinguishable with a non-negligible probability (by the means of the adversary $\mathcal{A}$ that may or may not break the `strong-security` in the respective experiments). Suppose f is a challenge oracle that is either a random function or a pseudorandom function with equal probability. Then, we can distinguish a random function from a pseudorandom one, via black-box access to f, with a non-negligible probability by letting $\langle L_i^{(j,0)}, L_i^{(j,1)}\rangle$ be obtained from an oracle query to f with respect to $L_i^{(j-1,1)}$. $\square$

6.5. *A concrete `strongly-secure` stateless group communication scheme*

Within the subset-cover revocation framework, [22] presented two concrete algorithms, namely the *complete subtree method* and the *subset difference method*. The difference between the two methods is how the collection of subsets (covering $\mathcal{N}\backslash\mathcal{R}$) is selected. Now we briefly review the *complete subtree method*, to which the above compiler is applicable.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 37

Suppose the receivers are the leaves in a rooted full binary tree with N leaves (assume that N is a power of 2). Such a tree contains $2N - 1$ nodes (leaves plus internal nodes) and for any $1 \leqslant i \leqslant 2N - 1$ we assume that v_i is a node in the tree. Denote by $ST(\mathcal{R})$ the unique (directed) Steiner Tree induced by the set $\mathcal{R}$ or vertices and the root; i.e., the minimal subtree of the full binary tree that connects all the leaves in $\mathcal{R}$. The collection of subsets $S_1, \ldots, S_w$ in this scheme corresponds to all complete subtrees in the full binary tree. For any node v_i in the full binary tree (either an internal node or a leaf, $2N - 1$ altogether) let subset S_i be the collection of receivers u that correspond to the leaves of the subtree rooted at node v_i. In other words, $u \in S_i$ iff v_i is an ancestor of u.

The **initialization** algorithm is simple: assign an *independent* and random key L_i to every node v_i in the complete tree, and provide every receiver u with the $\log N + 1$ keys associated with the nodes along the path from the root to leaf u. (As said before, if the secret information I_u is transmitted using a key established via a two-party authenticated key-exchange protocol, then the key is securely erased after the initialization.) The **broadcasting** algorithm is as follows. For a given set $\mathcal{R}$ of revoked receivers, let $u_1, \ldots, u_r$ be the leaves corresponding to the elements in $\mathcal{R}$. The method to partition $\mathcal{N} \setminus \mathcal{R}$ into disjoint subsets is as follows. Let $S_{i_1}, \ldots, S_{i_m}$ be all the subtrees of the original tree that "hang" off $ST(\mathcal{R})$; i.e., all subtrees whose roots $v_1, \ldots, v_m$ are adjacent to nodes of outdegree 1 in $ST(\mathcal{R})$, but are not in $ST(\mathcal{R})$. It follows immediately that this collection covers all nodes in $\mathcal{N} \setminus \mathcal{R}$ and only those. As a result, in the **decryption** algorithm, given a message

$$\langle [i_1, \ldots, i_m, E_{L_{i_1}}(K), \ldots, E_{L_{i_m}}(K)], F_K(M) \rangle$$

a receiver u needs to find whether any of its ancestors is among $i_1, \ldots, i_m$; note that there can be only one such ancestor, so u may belong to at most one subset.

Note that the number of subsets in a cover with N users and r revocations is at most $r \log(N/r)$. The message length is of at most $r \log(N/r)$ keys. Each receiver stores $\log N$ keys, and the center stores $2N - 1$ keys. The decryption process incurs $O(\log \log N)$ comparison operations (for finding the cover) plus two decryption operations.

Proof of the following theorem can be straightforwardly adapted from [22].

Theorem 6.3. *The complete subtree revocation scheme of the subset-cover framework is* `secure`.

As showed before, the subset-cover framework, and thus the complete subtree revocation scheme, is not `strongly secure`. As a corollary of Theorem 6.2 (which states that the compiler in Section 6.3 can transforms a `secure` stateless group communication scheme into a `strongly-secure` one) and Theorem 6.3 (which states that the above complete subtree method is a `secure` stateless group communication scheme), the scheme output by the compiler is `strongly-secure`.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 38

Corollary 6.1. *The stateless group communication scheme obtained by applying the compiler in* Section 6.3 *to the above* `secure` *complete subtree revocation scheme is* `strongly-secure`.

Now we analyze the *extra* complexities (corresponding to each revocation event) for achieving `strong-security`.

- The center updates its keys by evaluating $2N - 1$ pseudorandom functions (this corresponds to the worst case scenario that no keys have been corrupt – the corrupt keys, if known, do not need to be updated). Moreover, in order to encrypt a message, the center needs to evaluate $r \log(N/r)$ pseudorandom functions; this computational complexity can indeed be traded with an extra $2N - 1$ storage complexity. Since the center is typically powerful in terms of computation, communication, and storage, these extra complexities are insignificant.
- Each receiver needs to evaluate $2 \log N$ pseudorandom functions and at most stores $\log N$ keys. Even if the receivers are low-end equipment (e.g., sensors), these extra complexities should still be insignificant.

6.6. Discussions

The class of the stateless group communication schemes that can be made `strongly-secure` via the compiler in Section 6.3 should possess the following property: all the different keys belonging to $\{L_i\}_i \cup \{I_u\}_u$ are computationally-independent of each other. This explains why the above compiler applies to the *complete subtree method* of [22]. On the other hand, the *subset difference method* of [22], which does not achieve the desired `strong-security`, cannot made `strongly-secure` via the above compiler because the keys belonging to $\{L_i\}_i \cup \{I_u\}_u$ are not computationally independent.[4]

The stateless group communication schemes presented in [12], which outperforms [11,22] under certain interesting circumstances, are not `strongly-secure`. Unfortunately, they cannot be made `strongly-secure` via the above compiler for a similar reason. It is an interesting open question to make the stateless group communication schemes of [12] `strongly-secure` at an expense similar to the extra complexity imposed by the compilers presented in this paper.

7. Conclusion and open problems

We showed that a class of existing group communication schemes, stateful and stateless alike, are vulnerable to a realistic severe attack. We presented formal models that allow us to capture the desired security properties, and explore the relationships

[4] The independence condition can indeed be satisfied at the expense of each receiver storing $O(N)$ keys, which is clearly not scalable.

JCS ios2a v.2006/08/19 Prn:5/10/2006; 8:49 F:jcs274.tex; VTEX/VJ p. 39

between the security notions. We showed how some methods can make a *subclass* of existing schemes immune to the attack at a very small extra cost. An interesting open question is to make other schemes (e.g., the stateful [2,25] and the stateless [12]) secure against the attack without imposing any significant extra complexity.

Acknowledgements

We thank Jonathan Katz for illuminating discussions that led to the refined model in Section 3, our SASN'05 shepherd, Donggang Liu, for helpful feedback and communication, and the SASN'05 anonymous reviewers for useful comments. We thank the anonymous reviewers of this special issue for detailed and constructive sugges-

Join protocol for group-oriented rekeying: // suppose user u joins the group
server s generates a new key k_u for user u
server s finds a joining point x_j
server s attaches k_u to x_j
let x_0 be the root
$\bar{\mathsf{k}}_{j+1} \leftarrow k_u$
denote by x_{i-1} the parent of x_i for $1 \leqslant i \leqslant j$
let $\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_j$ be the current keys of $x_0, \ldots, x_j$, respectively
server s generates fresh keys $\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j$ // new keys of $x_0, \ldots, x_j$
$s \rightarrow \mathrm{USERSET}(\bar{\mathsf{k}}_0) : \{\hat{\mathsf{k}}_0\}_{\bar{\mathsf{k}}_0}, \{\hat{\mathsf{k}}_1\}_{\bar{\mathsf{k}}_1}, \ldots, \{\hat{\mathsf{k}}_j\}_{\bar{\mathsf{k}}_j}$
$s \rightarrow u : \{\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j\}_{k_u}$

Leave protocol for group-oriented rekeying: // suppose u leaves the group
let x_{j+1} be the deleted k-node for k_u
$\bar{\mathsf{k}}_{j+1} \leftarrow k_u$
server s finds the leaving point x_j (parent of k_u)
server s removes $\bar{\mathsf{k}}_{j+1}$ from the key tree
let x_0 be the root
denote by x_{i-1} the parent of x_i where $1 \leqslant i \leqslant j$
let $\bar{\mathsf{k}}_0, \bar{\mathsf{k}}_1, \ldots, \bar{\mathsf{k}}_j$ be the keys of $x_0, x_1, \ldots, x_j$ // they need to be changed
server s generates fresh keys $\hat{\mathsf{k}}_0, \hat{\mathsf{k}}_1, \ldots, \hat{\mathsf{k}}_j$ as the new keys of $x_0, x_1, \ldots, x_j$
FOR $i = 0$ TO j
let $\bar{\mathsf{k}}_{i_1}, \ldots \bar{\mathsf{k}}_{i_{z_i}}$ be the keys at the children of x_i in the new key tree
$L_i \leftarrow (\{\hat{\mathsf{k}}_i\}_{\bar{\mathsf{k}}_{i_1}}, \ldots, \{\hat{\mathsf{k}}_i\}_{\bar{\mathsf{k}}_{i_{z_i}}})$
$s \rightarrow \mathrm{USERSET}(\bar{\mathsf{k}}_0) \setminus \{u\} : (L_0, \ldots, L_j)$

Fig. 9. Join- and leave-incurred group-oriented rekeying in LKH.

tions that improved this paper. We thank Paul Parker for a careful proofreading that helped polish the writing.

This work was supported in part by ARO, NSF, and UTSA.

Appendix A. Join and Leave Protocols of LKH

For completeness, we briefly review join and leave protocols of LKH in Fig. 9. The notations are consistent with the main body of the paper.

References

[1] R. Anderson, On the forward security of digital signatures. Technical report, 1997.

[2] D. Balenson, D. McGrew and A. Sherman, *Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization*, Internet Engineering Task Force, Feb. 1999.

[3] M. Bellare and S. Miner, A forward-secure digital signature scheme, in: *Proc. CRYPTO 1999*, Lecture Notes in Computer Science No. 1666, Springer, 1999, pp. 431–448.

[4] M. Bellare and B. Yee, Forward-security in private-key cryptography, in: *Proc. Cryptographer's Track – RSA Conference (CT-RSA)*, Lecture Notes in Computer Science No. 2612, Springer, 2003, pp. 1–18.

[5] D. Boneh, G. Durfee and M. Franklin, Lower bounds for multicast message authentication, in: *Proc. EUROCRYPT 2001*, Lecture Notes in Computer Science No. 2045, Springer, 2002, pp. 437–452.

[6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, Multicast security: A taxonomy and some efficient constructions, in: *Proc. IEEE INFOCOM 1999*, 1999, pp. 708–716.

[7] R. Canetti, S. Halevi and J. Katz, A forward-secure public-key encryption scheme, in: *Proc. EUROCRYPT 2003*, Lecture Notes in Computer Science No. 2656, Springer, 2003, pp. 255–271.

[8] R. Canetti, T. Malkin and K. Nissim, Efficient communication-storage tradeoffs for multicast encryption, in: *Proc. EUROCRYPT 1999*, Lecture Notes in Computer Science No. 1592, Springer, 1999, pp. 459–474.

[9] A. Fiat and M. Naor, Broadcast encryption, in: *Proc. CRYPTO 1993*, Lecture Notes in Computer Science No. 773, Springer, 1993, pp. 480–491.

[10] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, *Journal of the ACM*, **33**(4) (1986), 792–807.

[11] D. Halevy and A. Shamir, The lsd broadcast encryption scheme, in: *Proc. CRYPTO 2002*, Lecture Notes in Computer Science No. 2442, Springer, 2002, pp. 47–60.

[12] N. Jho, J. Hwang, J. Cheon, M. Kim, D. Lee and E. Yoo, One-way chain based broadcast encryption schemes, in: *Proc. EUROCRYPT 2005*, Lecture Notes in Computer Science No. 3494, Springer, 2005, pp. 559–574.

[13] J. Katz and M. Yung, Complete characterization of security notions for probabilistic private-key encryption, in: *Proc. 32th ACM Symp. on Theory of Computing*, ACM, 2000, pp. 245–254.

[14] T. Kaya, G. Lin, G. Noubir and A. Yilmaz, Secure multicast groups on ad hoc networks, in: *Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2003)*, ACM Press, 2003, pp. 94–102.

[15] L. Lazos and R. Poovendran, Energy-aware secure multicast communication in ad-hoc networks using geographic location information, in: *Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2003)*, IEEE, 2003.

[16] L. Lazos and R. Poovendran, Cross-layer design for energy-efficient secure multicast communications in ad hoc networks, in: *Proc. IEEE International Conference on Communications (ICC 2004)*, IEEE, 2004.

[17] L. Lazos and R. Poovendran, Power proximity based key management for secure multicast in ad hoc networks, *ACM Journal on Wireless Networks (WINET)* (to appear).

[18] L. Lazos, J. Salido and R. Poovendran, Vp3: Using vertex path and power proximity for energy efficient key distribution, in: *Proc. IEEE VTC 2004*, IEEE, 2004 (invited paper).

[19] X. Li, Y. Yang, M. Gouda and S. Lam, Batch rekeying for secure group communications, in: *Proc. 10th International Conference on World Wide Web (WWW)*, ACM, 2001, pp. 525–534.

[20] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[21] D. Micciancio and S. Panjwani, Optimal communication complexity of generic multicast key distribution, in: *Proc. EUROCRYPT 2004*, Lecture Notes in Computer Science No. 3027, Springer, 2004, pp. 153–170.

[22] D. Naor, M. Naor and J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: *Proc. CRYPTO 2001*, J. Kilian, ed., Lecture Notes in Computer Science No. 2139, Springer, 2001, pp. 41–62.

[23] S. Rafaeli and D. Hutchison, A survey of key management for secure group communication, *ACM Computing Survey* **35**(3) (2003), 309–329.

[24] S. Setia, S. Koussih, S. Jajordia and E. Harder, Kronos: A scalable group re-keying approach for secure multicast, in: *Proc. IEEE Symposium on Security and Privacy*, IEEE, 2000, pp. 215–228.

[25] A. Sherman and D. McGrew, Key establishment in large dynamic groups using one-way function trees, *IEEE Trans. Softw. Eng.* **29**(5) (2003), 444–458.

[26] M. Waldvogel, G. Caronni, D. Sun, N. Weiler and B. Plattner, The VersaKey framework: Versatile group key management, *IEEE Journal on Selected Areas in Communications* **17**(9) (1999), 1614–1631.

[27] D. Wallner, E. Harder and R. Agee, Key management for multicast: Issues and architectures, Internet Draft, Sept. 1998.

[28] C. Wong, M. Gouda and S. Lam, Secure group communication using key graphs, in: *IEEE/ACM Transactions on Networking* (Preliminary version in *SIGCOMM 1998*), **8** (2000).

[29] S. Xu, On the security of group communication schemes based on symmetric key cryptosystems, in: *Proc. 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005)*, ACM, 2005, pp. 22–31.

[30] D. Yao, N. Fazio, Y. Dodis and A. Lysyanskaya, Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption, in: *Proc. 11th ACM conference on Computer and communications security (CCS 2004)*, ACM Press, 2004, pp. 354–363.

[31] S. Zhu, S. Setia, S. Xu and S. Jajodia, Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks, in: *Proc. 1st Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2004)*, IEEE Computer Society, 2004, pp. 42–51.